AFIT/GA/ENY/95D-02

Applications of Nonlinear Control Using the State-Dependent Riccati Equation

THESIS

David K. Parrish
Captain, USAF

AFIT/GA/ENY/95D-02

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

AFIT/GA/ENY/95D-02

Applications of Nonlinear Control Using the State-Dependent Riccati Equation

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

David K. Parrish, B.S.E., Aerospace Engineering

Captain, USAF

December, 1995

Acknowledgements

I would like to thank my reading committee, Dr. Canfield and Dr. Spenny, for their comments and suggestions, which have truly improved the quality of this thesis. I am also grateful to Dr. Hall and Dr. Wiesel for their assistance, especially since I know they were both busy with thesis students of their own. Most importantly I would like to express my thanks to my advisor Dr. Ridgely for his support and guidance. He has taught me a great deal about modern control and has motivated me to continue my education in this field.

I think I would be amiss to fail to acknowledge all the help my fellow students have also provided, from help with LaTeX to a multitude of other computer and control theory related questions. I have certainly enjoyed the friends I have made here at AFIT.

My last thanks is to my family. Although they are far away, they provide the love and support I need in the tasks that I undertake.

David K. Parrish

Table of Contents

List of Figures

Figure Page

List of Tables

Abstract

This thesis examines the relatively new theory of nonlinear control using state dependent coefficient factorizations to mimic linear state space systems. The control theory is a nonlinear quadratic approach, analagous to linear quadratic regulation. All implementations examined in this thesis are done strictly numerically.

This thesis is meant to provide a proof of concept for both satellite control and for an artificial pancreas to regulate blood glucose levels in diabetics by automatic insulin injection. These simulations represent only a first step towards practical use of the NQR method, and do not address noise rejection or robustness issues.

Applications of Nonlinear Control Using the State-Dependent Riccati Equation

I. Introduction

1.1 Overview

The majority of control work presently done is based on linear methods and analysis. For many dynamical systems, it is possible to linearize about a desired equilibrium and design a controller about that equilibrium. This is effective as long as the perturbations away from the equilibrium are small enough to be reasonably modelled by the linear system dynamics.

One of the most fundamental linear control synthesis methods is linear quadratic regulation (LQR). This method uses a trade-off between state deviations away from equilibrium and control usage by relative weighting of the states and controls. This method is the basis for least squares and Kalman filtering. Control usage is assumed to be a function of the states. Assume we have the following linear system in state space form

$$\dot{x} = Ax + Bu \tag{1.1}$$

where A and B are matrices and x and u are vectors. LQR assumes that all states are available to calculate the control needed. We then wish to choose our feedback control as

$$u = -Kx \tag{1.2}$$

The system now behaves as if the dynamics were

$$\dot{x} = (A - BK)x \tag{1.3}$$

which is stable as long as $(A - BK)$ has eigenvalues with negative real parts. Different choices of K with desired performance measures can be calculated using an algebraic Riccati equation (ARE) with various state and control weighting. There are many more advanced methods of linear control beyond LQR, but we will not be examining their nonlinear counterparts, if any, in this thesis.

Unfortunately, not all dynamical systems are handled well by linearized approximations. Since linear quadratic regulation is well understood and established, it is only natural to try to extend the LQR methods to regulate highly nonlinear systems. Based on the theory developed by Cloutier, D'Souza, and Mracek [CDM95], a nonlinear system can be factored into a form which mimics state space form. Analogous to linear methods there is a corresponding algebraic Riccati equation which is now a function of the states, rather than a constant as in LQR. The positive semidefinite solution of the Ricatti equation can be used to construct a stabilizing nonlinear feedback controller. This method will be referred to as nonlinear quadratic regulation (NQR).

The original intention of this thesis was to examine linear mixed-norm control methods for an artificial human pancreas. While attempting to linearize the biological dynamics of glucose and other hormones, it was found that the linearized dynamics did not capture the important dynamics of various states, the most important of these being glucose. For large external perturbations of the system, the glucose state's response was insignificant. Since it is the most important state that we wish to control through insulin injection, it appeared that linear control methods would not be adequate.

The nonlinear methods of Cloutier, et al. were then applied to the nonlinear pancreas dynamics to create a glucose state regulator, with encouraging results. Because of the complexity of the system, the control was implemented numerically. However, because of those complexities, simpler satellite dynamics were also examined to provide insight and validity to the numerical methods. While examining those simpler models, some valuable lessons were learned in numerical implementation, which will also be presented.

Presently, gain scheduling is commonly used as a method to extend linear control to systems which operate about many different equilibria. Using one or more measured quantities as scheduling variables, a different linear control is chosen for the appropriate values of the scheduling variables. From the artificial pancreas model we shall see that NQR can be implemented in a manner analogous to gain scheduling.

1.2 Objectives

The objectives of this thesis are twofold:

1. Examine numerical implementation of nonlinear quadratic regulation.

2. Provide a proof of concept using NQR on problems with practical application.

We will examine a numerical approach to get an initial feel for the applicability of NQR to certain problems. It is easily implemented in numerical simulations, plus alterations and permutations can be examined rather quickly. Analytical solutions for the problems we will examine can be quite complex, and with each new permutation investigated, a new solution would have to be found. The savings in time are more apparent when investigating nonlinear weights applied to the states and controls.

For the second objective, we will investigate one problem that has been getting increased attention lately. There is significant interest in trying to establish an automatic control system to regulate blood glucose levels in diabetics. Any such controller would be serving the role of an artificial pancreas. Linear controllers have not had significant success, partly due to the nonlinear nature of the dynamics. This thesis will examine a nonlinear controller, which could behave in a more natural manner than previously proposed nonlinear controllers.

This thesis also investigates NQR as applied to satellite control. Although there are many sufficient control schemes already in use, this investigation helps confirm the validity of the NQR

methodology, and could possibly provide more alternatives with different behaviors than current controllers.

1.3 Outline

Chapter II introduces the necessary theory from Cloutier, et al. In addition to the basic fundamentals of nonlinear quadratic regulation, optimality conditions and nonlinear state estimation theory are also presented for completeness. Because the implementation of NQR examined in this thesis is strictly numerical, Chapter III covers those numerical issues. It covers both how to choose a factorization suitable for numerical simulation, and how the simulations were implemented.

Chapters IV through VI examine nonlinear quadratic control of different satellite models. The dynamics are developed in Chapter IV for both externally and internally controlled satellites. Chapter V uses a basic satellite model to examine the issues in choosing different factorizations to represent the nonlinear system. Chapter VI looks at the applicability and effectiveness of NQR applied to more realistic satellite problems.

The next system examined is that of an artificial pancreas. There is current interest in developing automatic feedback controllers for the control of diabetes, which would both be safe and reduce the health risks associated with elevated glucose levels. Chapter VII presents the model developed by Naylor, Hodel, and Schumacher [NHS95]. The results of various implementation strategies are presented in Chapter VIII.

Conclusions and directions for follow-on work are given in Chapter IX. The appendices give additional information, including the MATLAB scripts of the nonlinear dynamics and controllers given in Appendix A. Appendix B presents the derivation on how the seven state externally controlled satellite model can be reduced to six states, to form a completely controllable problem.

II. Background Theory

This chapter presents the background theory of nonlinear regulation as developed by Cloutier, D'Souza, and Mracek [CDM95]. Specifically, the method involves finding a state-dependent coefficient (SDC) linear structure for which a stabilizing nonlinear feedback controller can be constructed. The following development is taken with only very minor modification from Cloutier, et al.

2.1 The Nonlinear Regulator Problem

We shall be considering the quadratic infinite-horizon cost function of the form

$$\text{minimize } J = \frac{1}{2}\int_{t_0}^{\infty} \left[x^T Q(x)x + u^T R(x)u\right] dt \tag{2.1}$$

subject to the nonlinear differential constraint

$$\dot{x} = f(x) + B(x)u \tag{2.2}$$

given state $x \in R^n$, control $u \in R^m, f(x) \in C^k, B(x) \in C^k$ and $Q(x) = H^T(x)H(x) \geq 0$, and $R(x) > 0$ for all x. We seek stabilizing solutions of the form

$$u = -K(x)x \tag{2.3}$$

which should be familiar from linear quadratic theory except that the matrices Q, R, and K all have elements that are functions of x.

2.2 State Dependent Coefficient Form

The constraint dynamics, Eqn. (2.2), can be written with a linear structure having state dependent coefficients

$$\dot{x} = A(x)x + B(x)u \tag{2.4}$$

so that

$$f(x) = A(x)x \tag{2.5}$$

The following definitions are associated with the SDC form:

Definition: $A(x)$ is an *observable parameterization* of the nonlinear system if the pair $\{H(x), A(x)\}$ is observable *for all* x.

Definition: $A(x)$ is a *controllable parameterization* of the nonlinear system if the pair $\{A(x), B(x)\}$ is controllable *for all* x.

Definition: $A(x)$ is a *detectable parameterization* of the nonlinear system if the pair $\{H(x), A(x)\}$ is detectable *for all* x.

Definition: $A(x)$ is a *stabilizable parameterization* of the nonlinear system if the pair $\{A(x), B(x)\}$ is stabilizable *for all* x.

2.3 State-Dependent Riccati Equation Technique

Associated with the nonlinear quadratic cost function is the state-dependent algebraic Riccati equation (SDARE):

$$A^T(x)P(x) + P(x)A(x) - P(x)B(x)R^{-1}(x)B^T(x)P(x) + Q(x) = 0 \tag{2.6}$$

Accepting only $P(x) \geq 0$, we can construct the nonlinear feedback control by

$$u = -R^{-1}(x)B^T(x)P(x)\,x \tag{2.7}$$

These equations can be solved analytically to produce an equation for each element of u, or solved numerically at a sufficiently high sampling rate.

The stability of the SDRE technique is given by the following theorem.

Stability Theorem. Given a detectable and stabilizable state dependent coefficient parameterization, the SDRE method has a closed loop solution which is *locally asymptotically stable*. For a proof, see [CDM95].

2.4 Optimality

Our performance index J is convex, so any stationary point is at least locally optimal. From our performance index and constrained dynamics we form the Hamiltonian function

$$H = \frac{1}{2}x^T Q(x)x + \frac{1}{2}u^T R(x)u + \lambda^T[A(x)x + B(x)u] \tag{2.8}$$

with stationary conditions

$$H_u = 0 \tag{2.9}$$

$$\dot{\lambda} = -H_x \tag{2.10}$$

$$\dot{x} = A(x)x + B(x)u \tag{2.11}$$

Using Eqs. (2.7) and (2.8) we have

$$H_u = R(x)u + B^T(x)\lambda \tag{2.12}$$

$$= \quad R(x)[-R^{-1}(x)B^T(x)P(x)x] + B^T(x)\lambda \tag{2.13}$$

$$= \quad B^T(x)[\lambda - P(x)x] \tag{2.14}$$

Thus, $H_u = 0$ if

$$\lambda = P(x)x \tag{2.15}$$

Satisfying Eqn. (2.15) for all time will satisfy the H_u optimality condition. From here we will drop the argument (x) notation for simplicity. Differentiating Eqn. (2.15) with respect to time gives

$$\dot{\lambda} = \dot{P}x + P\dot{x} \tag{2.16}$$

Using the optimality condition Eqn. (2.10) we also have

$$\dot{\lambda} = -Qx - \frac{1}{2}x^T Q_x x - \frac{1}{2}u^T R_x u - (x^T A_x^T + A^T + u^T B_x^T)\lambda \tag{2.17}$$

Equating Eqs. (2.16) and (2.17) with substitutions from Eqs. (2.2) and (2.7) gives

$$\dot{P}x + P(Ax - BR^{-1}B^T Px) = -Qx - \frac{1}{2}x^T Q_x x - \frac{1}{2}u^T R_x u - (x^T A_x^T + A^T + x^T PBR^{-1}B_x^T)Px \tag{2.18}$$

Rearrange to form

$$\dot{P}x + \frac{1}{2}x^T Q_x x + \frac{1}{2}u^T R_x u + x^T A_x^T Px - x^T PBR^{-1}B_x^T Px + [A^T P + PA - PBR^{-1}B^T P + Q]x = 0 \tag{2.19}$$

Furthermore, from Eqn. (2.6) note that the term in brackets is our SDARE, which equals zero, and substituting for u one more time, Eqn. (2.19) reduces to

$$\dot{P}x + \frac{1}{2}x^T Q_x x + \frac{1}{2}x^T PBR^{-1}R_x R^{-1}B^T Px + x^T A_x^T Px - x^T PBR^{-1}B_x^T Px = 0 \tag{2.20}$$

This is the SDRE *Optimality Criterion* which, if satisfied, guarantees the closed loop solution is locally optimal and may be the global optimum.

2.5 Solution to the SDARE Using a Hamiltonian Matrix

One method of finding the stabilizing solution to an algebraic Riccati equation involves the eigenvalues of an associated Hamiltonian matrix. The associated Hamiltonian matrix is given by

$$M \stackrel{d}{=} \begin{bmatrix} A(x) & -B(x)R^{-1}(x)B^T(x) \\ -Q(x) & -A^T(x) \end{bmatrix} \tag{2.21}$$

The Hamiltonian matrix M has dimension $2n \times 2n$, with the property that all its eigenvalues are symmetric about both the real and imaginary axes. A stabilizing solution exists only if M has n eigenvalues in the open left-half plane from whose corresponding eigenvectors a solution P can be found to Eqn. (2.6). If the n eigenvectors are used to form a $2n \times n$ matrix, and we denote the $n \times n$ square blocks as X and Y, so that

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \lambda_1 & \lambda_2 & \cdots & \lambda_n \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} = \begin{bmatrix} Y \\ X \end{bmatrix} \tag{2.22}$$

The solution to Eqn. (2.6) is then given by

$$P = XY^{-1} \tag{2.23}$$

An excellent reference is Zhou, et al. [ZDG95], which gives more detailed developments and proofs for solving various forms of the algebraic Riccati equation.

For some of our problems we will be interested in solutions where, because of our parameterization, n eigenvalues with negative real parts might not be available. We shall see in the satellite dynamics section how a certain parameterization will guarantee zero eigenvalues. If this is a result of an algebraic constraint where all the states cannot be driven to zero, and we can remove this state from our cost function J, then we can still construct a "stabilizing" controller.

If there are $m < n$ left half plane eigenvalues and $n - m$ zero eigenvalues with at least $\frac{n-m}{2}$ corresponding linearly independent eigenvectors, then we can construct the $2n \times n$ matrix of Eqn. (2.22) using the m eigenvectors of left half plane eigenvalues and the independent $\frac{n-m}{2}$ eigenvectors associated with the zero eigenvalues. The usable solution P is still given by Eqn. (2.23). This solution is known as a neutrally stabilizing solution.

2.6 Nonlinear State Estimation

Analagous to linear methods, Mracek, Cloutier, and D'Souza [MCD95] have also developed theory for a nonlinear state estimator. Using the dual formulation to the nonlinear quadratic regulator problem, a nonlinear estimator can formed. The development for this section was taken from [MCD95].

Assuming our measurement is a nonlinear function of x such that

$$y = g(x) \tag{2.24}$$

we need to form a state dependent coefficient measurement

$$y = C(x)x \tag{2.25}$$

For the optimal estimation problem, we will use a cost function of the form

$$\underset{\hat{x}}{\text{minimize}} \quad J = \frac{1}{2}E\left[\int_{t_0}^{\infty}\left[(x-\hat{x})^T\Gamma^T W^{-1}\Gamma(x-\hat{x}) + (y-C\hat{x})^T V^{-1}(y-C\hat{x})\right]dt\right] \tag{2.26}$$

subject to the nonlinear differential constraints

$$\dot{x} = A(x)x + \Gamma w \tag{2.27}$$

$$y = C(x)x + v \tag{2.28}$$

where $W = E[w^T w]$, the variance of the process noise, and $V = E[v^T v]$, the variance of the measurement noise.

Associated with the SDC measurement form we have the following definitions:

Definition: $A(x)$ and $C(x)$ form an *observable parameterization* of the nonlinear system if the pair $\{C(x), A(x)\}$ is observable *for all* x.

Definition: $A(x)$ is a *controllable parameterization* of the nonlinear system if the pair $\{A(x), \Gamma\}$ is controllable *for all* x.

Definition: $A(x)$ and $C(x)$ form a *detectable parameterization* of the nonlinear system if the pair $\{C(x), A(x)\}$ is detectable *for all* x.

Definition: $A(x)$ is a *stabilizable parameterization* of the nonlinear system if the pair $\{A(x), \Gamma\}$ is stabilizable *for all* x.

Using the dual of the regulator problem, the nonlinear estimator is given by

$$\frac{\mathrm{d}\hat{x}}{\mathrm{d}t} = A(\hat{x})\hat{x} + K_f(y_m - \hat{y}) \tag{2.29}$$

where

$$\hat{y} = C(\hat{x})\hat{x} \tag{2.30}$$

$$K_f = Y(x)C^T(\hat{x})V^{-1} \tag{2.31}$$

and $Y(x)$ is the positive semidefinite solution to

$$A(\hat{x})Y(x) + Y(x)A^T(\hat{x}) - Y(x)C^T(\hat{x})V^{-1}C(\hat{x})Y(x) + \Gamma^T W\Gamma = 0 \tag{2.32}$$

The estimator will not be optimal unless a time dependent parameterization meeting the optimality condition is used. However, not requiring optimality may still result in a sufficient estimator.

III. Numerical Approach To SDARE Control

3.1 State-Dependent Coefficient Factorization

3.1.1 Controllable Parameterization. To solve the SDARE by numerical methods, a controllable factorization must first be found. For example, take a two state problem with the form

$$\begin{aligned} \dot{x}_1 &= f(x)x_1 + \mathrm{b}u \\ \dot{x}_2 &= x_1 x_2 \end{aligned} \tag{3.1}$$

One factorization would be

$$\dot{\mathrm{x}} = \begin{bmatrix} f(x) & 0 \\ x_2 & 0 \end{bmatrix} x + \begin{bmatrix} \mathrm{b} \\ 0 \end{bmatrix} u \tag{3.2}$$

and a second factorization would be

$$\dot{\mathrm{x}} = \begin{bmatrix} f(x) & 0 \\ 0 & x_1 \end{bmatrix} x + \begin{bmatrix} \mathrm{b} \\ 0 \end{bmatrix} u \tag{3.3}$$

The first factorization is controllable whereas the second is not. State x_1 is dependent directly on the control input, and state x_2 is in turn dependent on x_1. In the first factorization this information is maintained. The (2,1) term in the A matrix shows the coupling between these states. The second factorization has hidden the cross coupling information. Even though the (2,2) term is indeed a function of x_1, the pointwise linear representation of the system does not know how that term is changing and thus is unable to control x_2. This is easily seen by examining the controllability matrix for each system. The first system has rank two and the second has only rank one, provided that $f(x)$ and x_2 are not simultaneously equal to zero.

3.1.2 Optimal Parameterizations. Different factorizations of the same problem can have different control histories. When examining a fixed parameterization to solve the SDARE, any given SDC choice will not necessarily be optimal. To establish optimality, Cloutier, et al. establish a parameterization set which spans a hyperplane of possible solution parameterizations using spanning variables $\{\alpha_i\}$. To achieve this optimality, each $\{\alpha_i\}$ must be solved for as a function of time which involves solving a two point boundary value problem.

However, a numerical approach using a fixed suboptimal SDC choice can provide insight into increasingly complex systems for which finding the positive definite solution with associated boundary conditions is not a trivial undertaking. It is not implied that any factorization in the following chapters is the optimal solution. Chapter V will show variations in control usage between two parameterizations with equal weighting functions. One parameterization might be more nearly optimal than the other, but it might just take changing the Q and R weightings to create a practical solution. Also, from an implementation point of view, optimality might not be significant. Any stabilizable and detectable parameterization will achieve the same final result, i.e. perturbed states will return to zero. If the state or control deviations or settling time are unacceptable no matter what the Q and R weights are, then pursuing an optimal solution will most likely be necessary.

3.1.3 Ill-Conditioned Parameterizations. Numerical problems can arise when one state differential equation is parameterized in a way which is nearly the same as another, i.e., a multiple of the other. This is best seen using a real example, which will be illustrated in Section 6.3.1, Eqn. (6.10). Further discussion of this issue will be deferred to that section.

3.2 Numerical Implementation

All simulations in this thesis were accomplished using MATLAB and SIMULINK [MAT]. One function contained the nonlinear differential equations. A second function created the A, B, Q, and R matrices of the SDC parameterizations. At each time step, the state dependent algebraic

Riccati equation

$$A^T(x)P(x) + P(x)A(x) - P(x)B(x)R^{-1}(x)B^T(x)P(x) + Q(x) = 0 \tag{3.4}$$

was solved for a stabilizing or neutrally stabilizing P. The instantaneous control u was calculated by

$$u = -R^{-1}(x)B^T(x)P(x)\,x \tag{3.5}$$

and fed back to the nonlinear dynamics. All of these calculations took place in the function labeled "nonlinear controller". The following figure shows the SIMULINK representation.

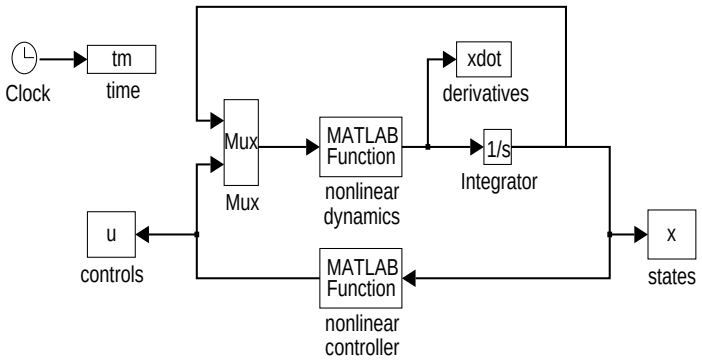

Figure 3.1 SIMULINK NQR Controller Model

To calculate P, the MATLAB algebraic Riccati equation solver had to be modified to allow for zero eigenvalues of the Hamiltonian matrix as discussed in Section 2.5. This modified routine was invoked in MATLAB in the the form SDARE(A, B*inv(R)*B', Q). The function SDARE will be discussed further in the next section.

3.3 Numerical Issues

3.3.1 Zero Eigenvalue Pairs. An interesting problem arose in the simulations of satellite attitude control. Originally set up using Euler angles, it was suggested by Hall [Hal95a] that they be examined using quaternions since that is the more standard coordinate system. Because quaternions describe 3 space with 4 parameters, there is an inherent constraint. Due to this constraint, the associated Hamiltonian of the SDARE will have a zero eigenvalue pair, because of symmetry, implying a nonstabilizable mode, i.e. the constraint itself. However, if the nature of this nonstabilizable mode is something we can neglect, we can construct a neutrally stabilizing solution and use it.

Our goal in nonlinear quadratic regulation is to drive the state deviations and control usage to zero. If we have a constrained state as above, it may only be possible to drive all but one state to zero. The last state will settle at a non-zero value, consistent with its zero eigenvalue. This state should be left unweighted and therefore undetectable. This prevents its inclusion in the cost function, which would otherwise be infinite. If the value the unconstrained state will settle at is known and acceptable, then the neutrally stabilizing solution to the SDARE is acceptable.

This can result in fairly effective controllers; however, there is one precaution. If there is more than one solution to the constrained states, there is no guarantee that the unweighted state will go to the desired solution. As a regulator this could pose a problem when a disturbance is large enough to move the states into the vicinity of the other solutions. In this sense the system is not completely controllable to the desired equilibrium. For example, take a two state vector that is constrained to have magnitude of 1, and you wish to keep it in an equilibrium value of $[0\ 1]^T$ by penalizing only the first state. Starting from equilibrium the controller might work fine. If however, the states are disturbed to a value close to $[0\ -1]^T$, the controller will probably then drive the first state to zero leaving the second state at its new value. The system will now be regulated about this undesired equilibrium.

The standard MATLAB algebraic Riccati equation solver is designed to return only stabilizing solutions. Because we are now interested in neutrally stabilizing solutions as well, the function ARE(a,b,c) was altered into a new function SDARE(a,b,c). This was done by eliminating the error checking routine which checked for n negative eigenvalues. The eigenvectors are sorted by their respective eigenvalue sign from negative to zero and then positive. The modified routine would now return a neutrally stabilizing solution because it used the eigenvector associated with an imaginary axis eigenvalue. Admittedly, this is not a robust error checking method, but it worked for the satellite examples examined. More coding and error checking would have to be done for the routine to be used on any example. The pancreas dynamics required only the standard ARE solver.

3.3.2 Singularity. A problem can be factored into a form where there is division by any one of the states. This is numerically disastrous as the states approach zero. For a problem as complex as the pancreas in Chapter VII, this factorization may be unavoidable. If so, it may be necessary to introduce a dead band where the states are nearly zero and no control is used since the states are sufficiently close to zero. This deadband was introduced into the controller for the artificial pancreas, because of division by the glucose state. The numerics were well behaved for very small values of the glucose state, so a deadband for values less than 0.0001 only was used. This is sufficiently close to our desired value and results in no performance loss.

IV. Satellite Dynamics

This chapter gives the simplified satellite dynamics for the problems considered in this thesis, and then incorporates those dynamics into a state dependent coefficient form.

The notation $\|v\|$ will be used to denote the Euclidean norm of a vector; i.e., its magnitude.

4.1 Rigid Body with 3-axis External Torques

Assuming a rigid body, a satellite controlled with external torques obeys Euler's equation given by

$$\dot{\omega} = -J^{-1}\omega \times J\omega + J^{-1}T \tag{4.1}$$

where ω is the body axis angular velocity vector, J is the inertia tensor, and T is the external torques about each body axis. The derivation of rigid body motion can be found in most dynamics texts. A good reference is Chobotov [Cho91], which concentrates specifically on satellites.

4.2 Rigid Body with Internal Stabilizing Rotors

Another interesting problem we will examine for which nonlinear quadratic regulation could have practical application is a satellite stabilized by internal spinning rotors. The internal rotors provide not only stability, but by changing the angular rates of each rotor, the satellite can be brought to a new orientation. The total angular momentum of the satellite will be constant since no external forces are present.

We will define the vector μ, containing three states, to have elements representing the axial angular momentum of each rotor relative to inertial space. The vector x, also containing three states, is the angular momentum expressed in body axes and is scaled such that $\|x\| = 1$. The control vector u will be the torque applied to each rotor. The system dynamics are

$$\dot{\mu} = u \tag{4.2}$$

$$\dot{x} = x \times J^{-1}(x - A\mu) \tag{4.3}$$

Given n rotors (controllers), then $\mu \in R^n$, $u \in R^n$, $x \in R^3$, $J \in R^{3\times3}$, and $A \in R^{3\times n}$. J is a scaled inertia tensor given by

$$J = J_{\text{sat}} - A^T J_{\text{rot}} A \tag{4.4}$$

where J_{sat} is the satellite inertia tensor, and J_{rot} is a diagonal $n \times n$ matrix with elements corresponding to each rotor's inertia. The matrix A has columns of unit vectors representing the direction of each rotor's axis of spin in body axes, and whose order corresponds to the order of J_{rot}. The equations of motion for this section were developed by Hall [Hal95b]. Leaving the notation the same unfortunately leads to another use of A and x. Since these variables will appear as elements of $A(x)x$ their meaning should be taken from context.

We will also need the angular rates, ω, given by

$$\omega = J^{-1}(x - A\mu) \tag{4.5}$$

Notice the satellite is stationary when $x = A\mu$. Also note that Eqn. (4.3) can be simply written as $\dot{x} = x \times \omega$.

4.3 Attitude Coordinates

The above sections give the equations for body axis angular rates. If in addition to angular velocities we want to regulate an inertial position, then we must also include an inertial coordinate system. To express the orientation of the satellite there are several options. Euler angles are a common representation, but because the equation of motions can become singular for certain

angles, quaternions are a preferable coordinate system. Initial quaternion values can be found from the initial rotation matrix.

Quaternion dynamics will be included in the SDC parameterization to regulate inertial attitude of the satellite. There are four quaternions whose dynamics are related to the body axis angular rates by

$$\begin{aligned} \dot{q}_1 &= 0.5\,(q_2\omega_3 - q_3\omega_2 + q_4\omega_1) \\ \dot{q}_2 &= 0.5\,(-q_1\omega_3 + q_3\omega_1 + q_4\omega_2) \\ \dot{q}_3 &= 0.5\,(q_1\omega_2 - q_2\omega_1 + q_4\omega_3) \\ \dot{q}_4 &= 0.5\,(-q_1\omega_1 - q_2\omega_2 - q_3\omega_3) \end{aligned} \tag{4.6}$$

If we adopt a vector notation of $q = [q_1\ q_2\ q_3\ q_4]^T$ we can use a shorthand notation of

$$\dot{q} = \mathcal{Q}\omega \tag{4.7}$$

where the matrix $\mathcal{Q}$ is given by

$$\mathcal{Q} \stackrel{d}{=} 0.5 \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \tag{4.8}$$

Quaternions also have the property that $\|q\| = 1$. This means in the regulator problem that only three quaternions can be driven to zero. The fourth will go to ± 1. Fortunately, coordinates of $[0\ 0\ 0\ 1]^T$ and $[0\ 0\ 0\ -1]^T$ represent the same spatial orientation as do any $\pm q$. This means regulating any three quaternions to 0 results in a unique solution.

As we shall see later, the properties of quaternions result in a rank defective controllability matrix. This would appear to cause problems; however, we will be able to use the neutrally stabilizing solution to the SDARE as discussed previously.

There are several different ways quaternions can be defined. The quaternions used in this thesis are the same as found in Chobotov [Cho91].

4.4 Cross Product Notation

One notation needs to be introduced to simplify later representations. We will define

$$x^{\times} \stackrel{d}{=} \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{4.9}$$

which will allow the vector cross product $x \times y$ to be denoted as $x^{\times}y$, now a matrix times a vector.

V. Suboptimal SDC Controllers

This chapter provides a conceptual feel for both how to choose a fixed SDC parameterization, and how optimality can impact performance.

5.1 SDC Parameterization

We are interested in seeing how different controllable parameterizations will behave, to see if optimality is always a concern. If we assume an inertia tensor in principle axes, with inertias J_1, J_2, and J_3, we can expand and write the dynamics of Section 4.1 as the following three equations:

$$\begin{aligned} \dot{\omega}_1 &= \frac{J_2 - J_3}{J_1}\omega_2\omega_3 + \frac{T_1}{J_1} \\ \dot{\omega}_2 &= \frac{J_3 - J_1}{J_2}\omega_1\omega_3 + \frac{T_2}{J_2} \\ \dot{\omega}_3 &= \frac{J_1 - J_2}{J_3}\omega_1\omega_2 + \frac{T_3}{J_3} \end{aligned} \tag{5.1}$$

These equations will be parameterized into four different SDC forms. The first will be

$$\dot{\omega} = \begin{bmatrix} 0 & \frac{J_2-J_3}{J_1}\cdot\omega_3 & 0 \\ \frac{J_3-J_1}{J_2}\cdot\omega_3 & 0 & 0 \\ 0 & \frac{J_1-J_2}{J_3}\cdot\omega_1 & 0 \end{bmatrix}\omega + \begin{bmatrix} \frac{1}{J_1} & 0 & 0 \\ 0 & \frac{1}{J_2} & 0 \\ 0 & 0 & \frac{1}{J_3} \end{bmatrix} T \tag{5.2}$$

which will be referred to as parameterization A (PA). Parameterization B (PB) will be similar except we will change the factorization of the first equation. Parameterization B has the following

SDC form

$$\dot{\omega} = \begin{bmatrix} 0 & 0 & \frac{J_2-J_3}{J_1}\cdot\omega_2 \\ \frac{J_3-J_1}{J_2}\cdot\omega_3 & 0 & 0 \\ 0 & \frac{J_1-J_2}{J_3}\cdot\omega_1 & 0 \end{bmatrix} \omega + \begin{bmatrix} \frac{1}{J_1} & 0 & 0 \\ 0 & \frac{1}{J_2} & 0 \\ 0 & 0 & \frac{1}{J_3} \end{bmatrix} T \tag{5.3}$$

We can add elements to PB by adding and subtracting terms from each state equation. In this manner we can create an A matrix with no empty elements. This altered parameterization will be called PB2, and is given by

$$\dot{\omega} = \begin{bmatrix} \omega_2 & -\omega_1 & \frac{J_2-J_3}{J_1}\cdot\omega_2 \\ \frac{J_3-J_1}{J_2}\cdot\omega_3 & \omega_3 & -\omega_2 \\ -\omega_3 & \frac{J_1-J_2}{J_3}\cdot\omega_1 & \omega_1 \end{bmatrix} \omega + \begin{bmatrix} \frac{1}{J_1} & 0 & 0 \\ 0 & \frac{1}{J_2} & 0 \\ 0 & 0 & \frac{1}{J_3} \end{bmatrix} T \tag{5.4}$$

The last parameterization will have the states factored more evenly. This will be parameterization C (PC), given by

$$\dot{\omega} = \begin{bmatrix} 0 & \frac{J_2-J_3}{2J_1}\cdot\omega_3 & \frac{J_2-J_3}{2J_1}\cdot\omega_2 \\ \frac{J_3-J_1}{2J_2}\cdot\omega_3 & 0 & \frac{J_3-J_1}{2J_2}\cdot\omega_1 \\ \frac{J_1-J_2}{2J_3}\cdot\omega_2 & \frac{J_1-J_2}{2J_3}\cdot\omega_1 & 0 \end{bmatrix} \omega + \begin{bmatrix} \frac{1}{J_1} & 0 & 0 \\ 0 & \frac{1}{J_2} & 0 \\ 0 & 0 & \frac{1}{J_3} \end{bmatrix} T \tag{5.5}$$

Conceptually we can see that all four parameterizations will probably behave differently in some fashion. At any point in time, the matrix A has a definite form with values dependent on the state ω. Each parameterization above, although representing the same dynamics, looks like a different linear system at each point in time. For example, in PA the algebraic Riccati solver will see a linear system where ω_3 is coupled to only one other state, whereas for PB2 it will see coupling of each state to every other state. These will indeed give different control laws based on the different parameterizations.

5.2 Results

Now let's examine how each model performs in a numerical simulation. We will use an inertia matrix of

$$J = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 20 \end{bmatrix} \tag{5.6}$$

All control simulations for this chapter will have initial angular velocities of

$$\omega_0 = \begin{bmatrix} 4 \\ 4 \\ 2 \end{bmatrix} \tag{5.7}$$

To compare all four factorizations we will use a state penalty weight of $Q = 10I_{3\times3}$ and a control penalty weight of $R = I_{3\times3}$. The results of parameterization A can be seen in Figures 5.1 and 5.2. The controller works quite well and brings the satellite to rest in around 35 seconds. We can compare this time history with the results of the PB model shown in Figures 5.3 and 5.4. Changing only two elements of the A matrix resulted in maximum control magnitudes that more than doubled, even though the second controller still achieves the same final results and settles in nearly the same time. The differences in state deviations are not as severe, but PA is a better regulator.

Next, compare the PB2 results shown in Figures 5.5 and 5.6 with the previous examples. Here we see that adding terms which don't change the dynamics can have a very adverse effect on control usage. The control usage has magnitude almost five times the original PA controller, while the settling time is still roughly the same. This controller does not initially regulate the states well at all.

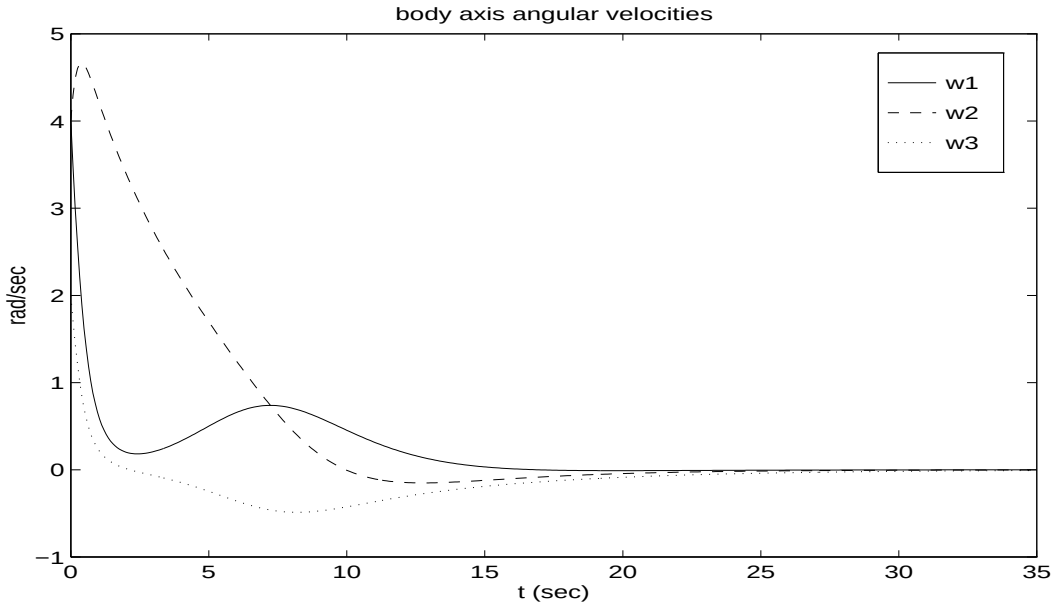

Figure 5.1 PA with $Q = 10R$: State Deviations

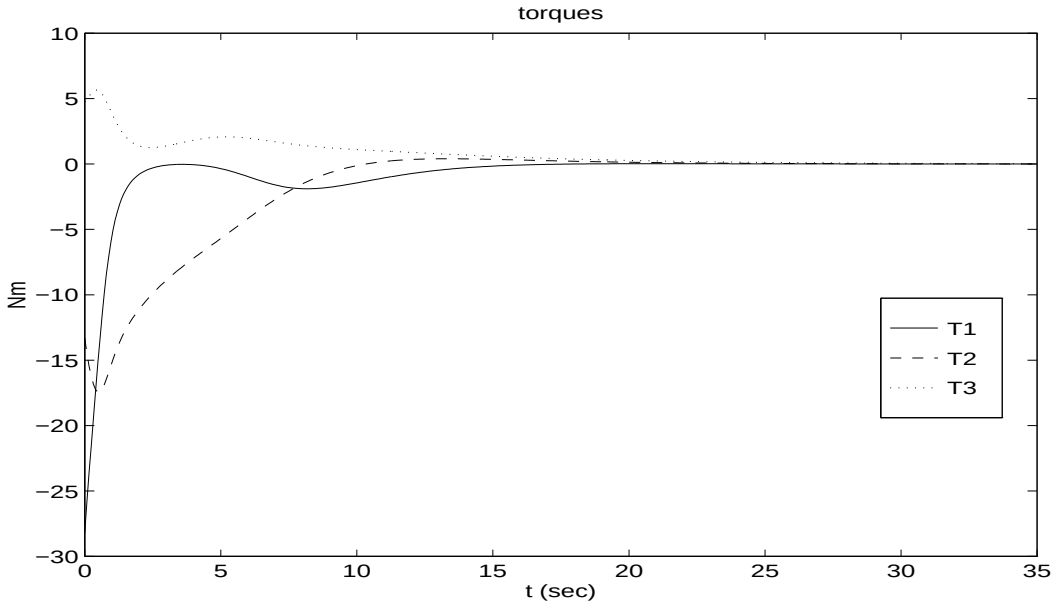

Figure 5.2 PA with $Q = 10R$: Control History

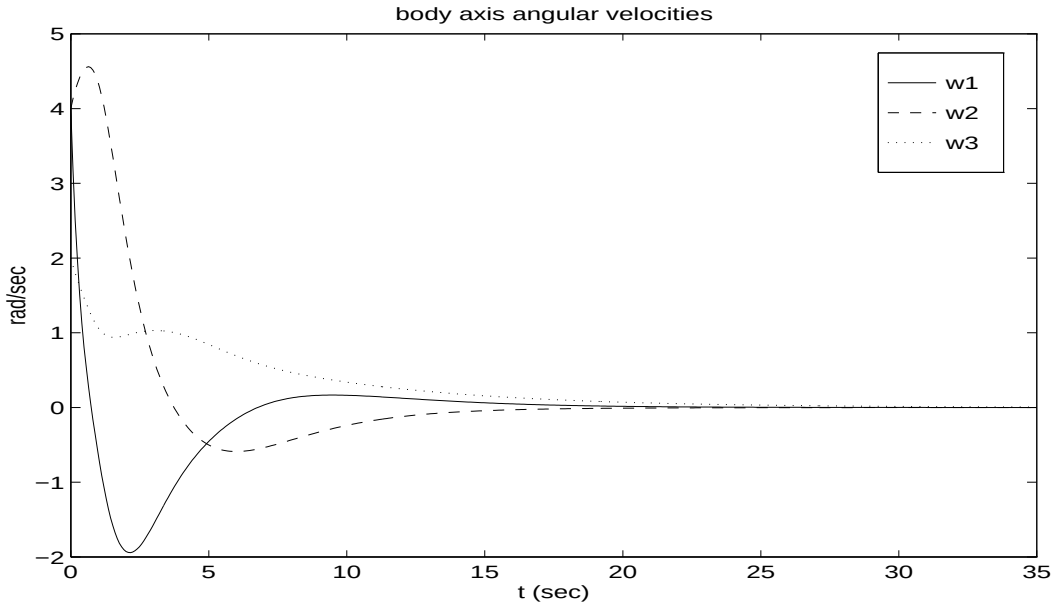

Figure 5.3 PB with $Q = 10R$: State Deviations

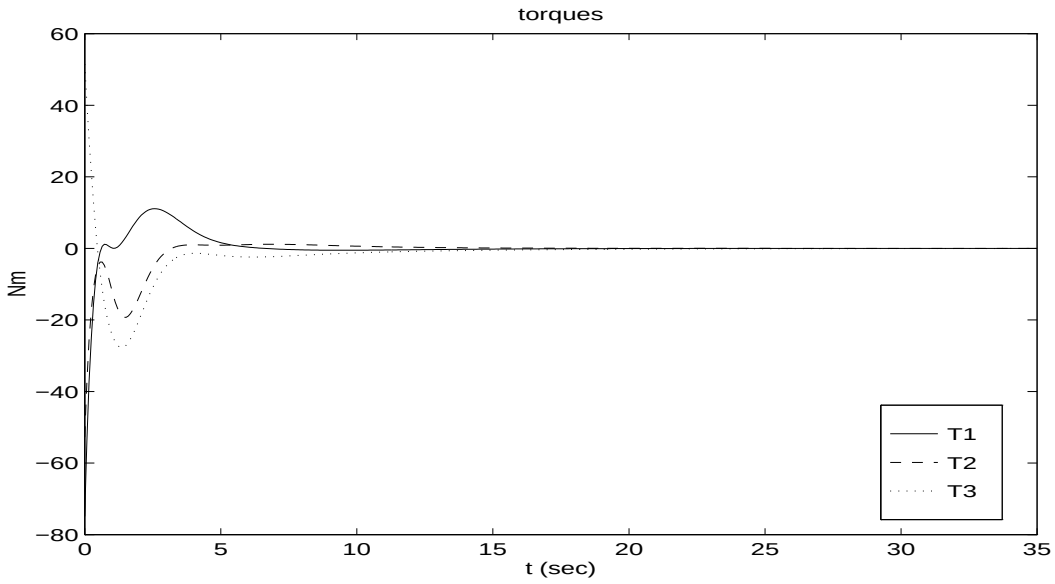

Figure 5.4 PB with $Q = 10R$: Control History

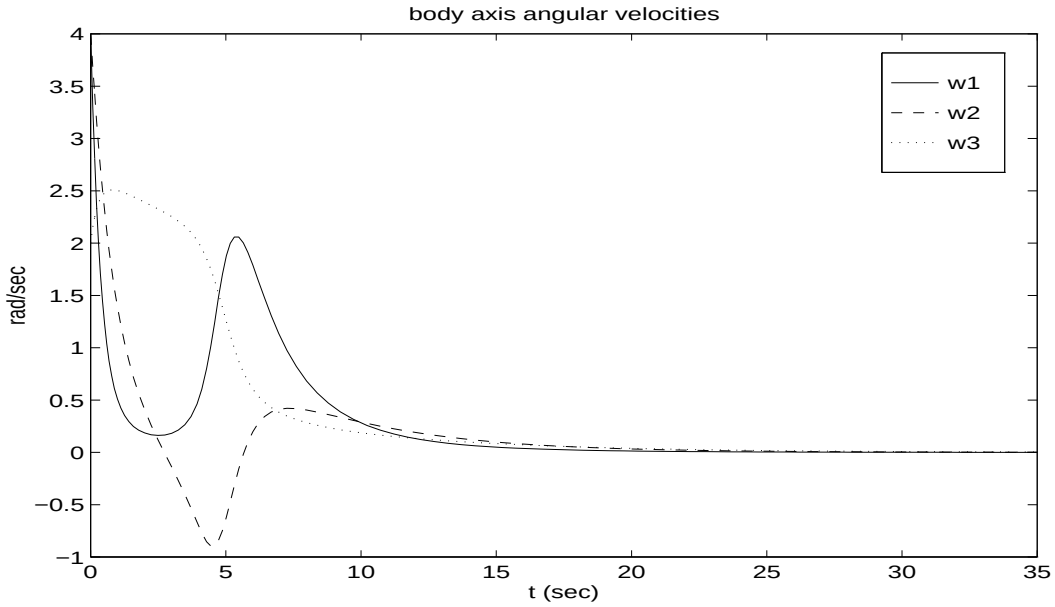

Figure 5.5 PB2 with $Q = 10R$: State Deviations

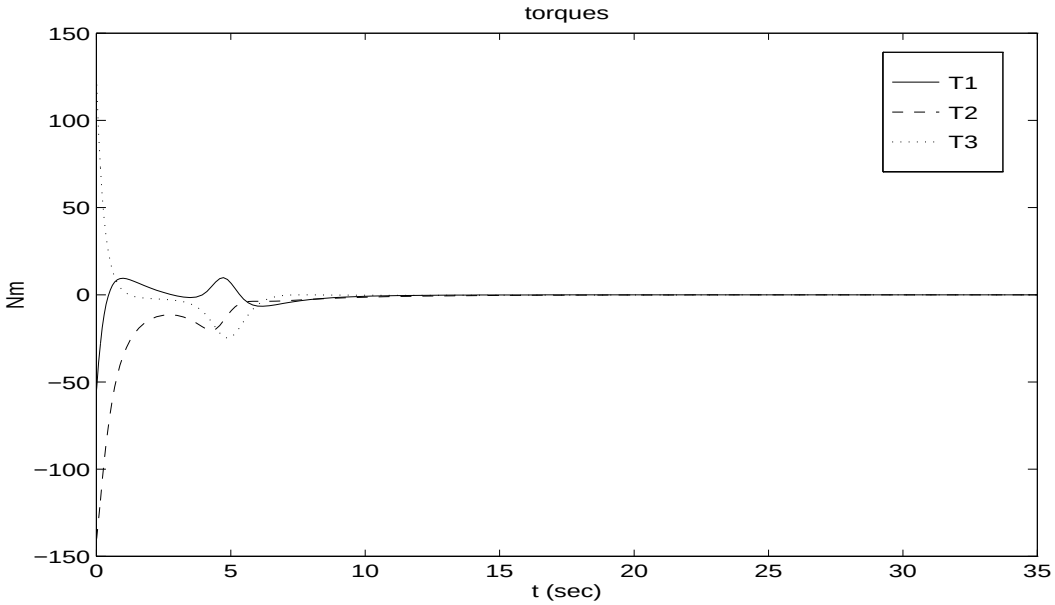

Figure 5.6 PB2 with $Q = 10R$: Control History

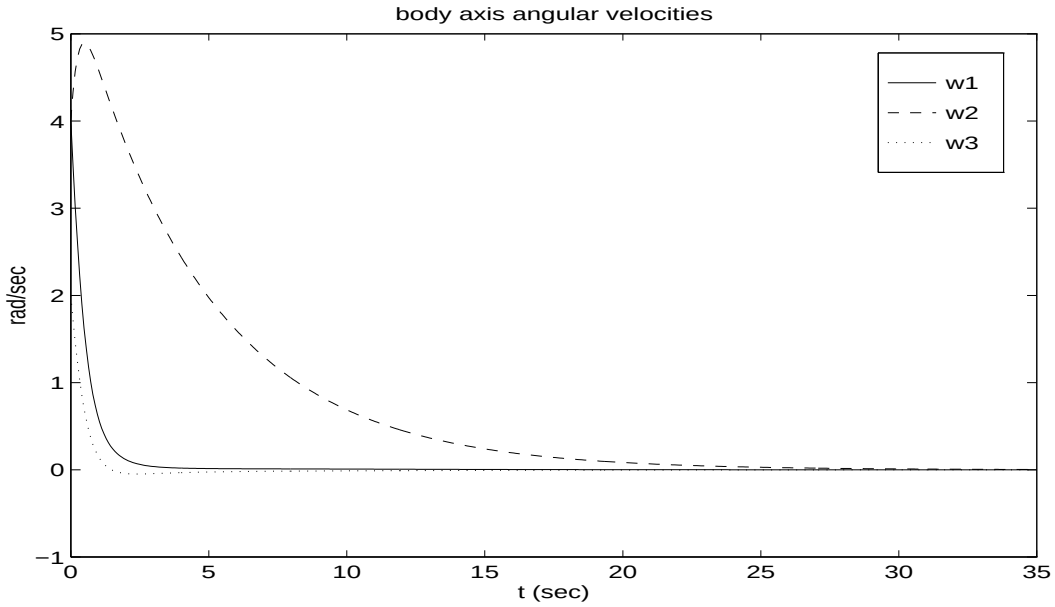

Figure 5.7 PC with $Q = 10R$: State Deviations

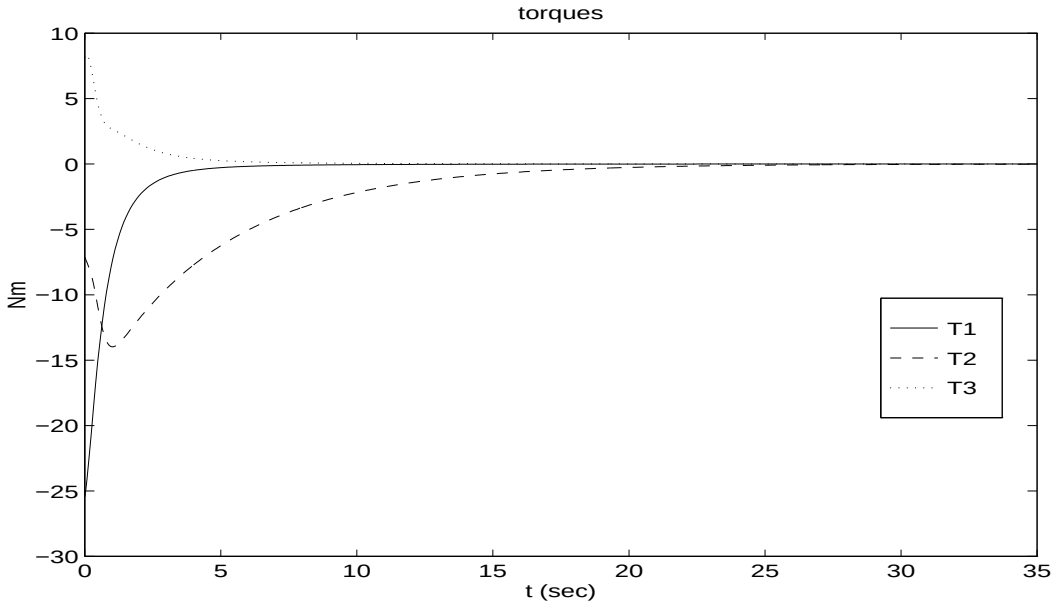

Figure 5.8 PC with $Q = 10R$: Control History

The last SDC form, PC, gives a controller with the results seen in Figures 5.7 and 5.8. This controller produces considerably smoother time histories with maximum control magnitudes less than all previous SDC forms. The first and third angular velocities are regulated very quickly; the second is much slower, but still settles in the same amount of time as the previous controllers. This is an excellent regulator, especially if ω_2 is not as critical. Increasing state weighting will bring ω_2 back to equilibrium quicker, but at the cost of more control usage.

We can evaluate the cost function associated with each parameterization using numerical integration. Using a trapezoidal approximation, the cost functions were $1.71e+3$, $3.35e+3$, $1.19e+4$, and $1.73e+3$ respectively. Since parameterizations A and C had the closest performance of the four forms examined, we will now see how different weightings affect these two controllers. The next simulations will have state weighting of $Q = I$ and control weighting of $R = 10I$.

Figures 5.9 and 5.10 show PA's performance, and PC's performance is shown in Figures 5.11 and 5.12. Here the performance is considerably different. Parameterization A has a more rapidly changing control usage. The state deviations and control usage are oscillatory as compared to the still exponential nature of parameterization C. Time to settle for both controllers is around five minutes. Parameterization A now has a cost function of $2.07e+3$ while PC's cost function is $4.46e+3$. Here we can see that although one parameterization is more optimal than another, it may not necessarily have desirable characteristics.

Figures 5.13 and 5.14 show parameterization C with an even larger control penalty of $R = 10^5 I$. Notice that ω_1 and ω_3 still have the same behavior, while ω_2 is directly related to the control weighting. Parameterization C tries to stop the tumbling motion of the satellite as quickly as possible, and then brings the third state to rest at a rate directly related to control weighting.

We will examine one last simulation. Since parameterization C has maximum control magnitudes that are almost invariant to changes in the penalty matrices Q and R, we will see if limiting control usage still results in a stable controller. Figures 5.15 and 5.16 show the results of parameter-

ization C using the original weights of the first example, except that control usage is sent through a limiter of ± 2 Nm. Notice that the controller uses maximum control until the state deviations are reduced to a level where the control exponentially dies out. The controller is still quite effective even though the reduction in control usage could not be accomplished by control weighting.

5.3 Summary

We have seen that differences between factorizations produce different state and control histories, and can be very sensitive to state and control weightings. For the examples considered, heavy control penalties greatly enlarged the differences between parameterizations A and C, which had relatively similar performance under heavy state weighting. We also saw that although relatively little changes in the A matrix could effect the control and state histories, there was little change in final settling time.

Adding elements to the A matrix so that dynamics cancel out is a trick used by Cloutier, et al., [CDM95] for a system that has a zero A matrix. In the example here, it was seen that there were adverse effects from adding those terms to an A matrix that was already well behaved. Adding terms should probably only be done to handle such special cases like the one examined in [CDM95].

Conceptually, we have learned a little about forming a good state dependent coefficient factorization for a given system. Of the four parameterizations examined, parameterization C appears to give the overall better performance. It differs from the other parameterizations in that its dynamics are more evenly distributed among all the terms. State Dependent Coefficient form is a way to make a nonlinear system look linear at each point in time. By distributing the dynamics, the linear representation shows all the coupling between states and their relationships to one another. Therefore, a nonlinear term like $x_1 x_2 x_3$ might be better represented by dividing it in thirds and factoring out each state, rather than factoring one state and leaving the rest as a single coefficient.

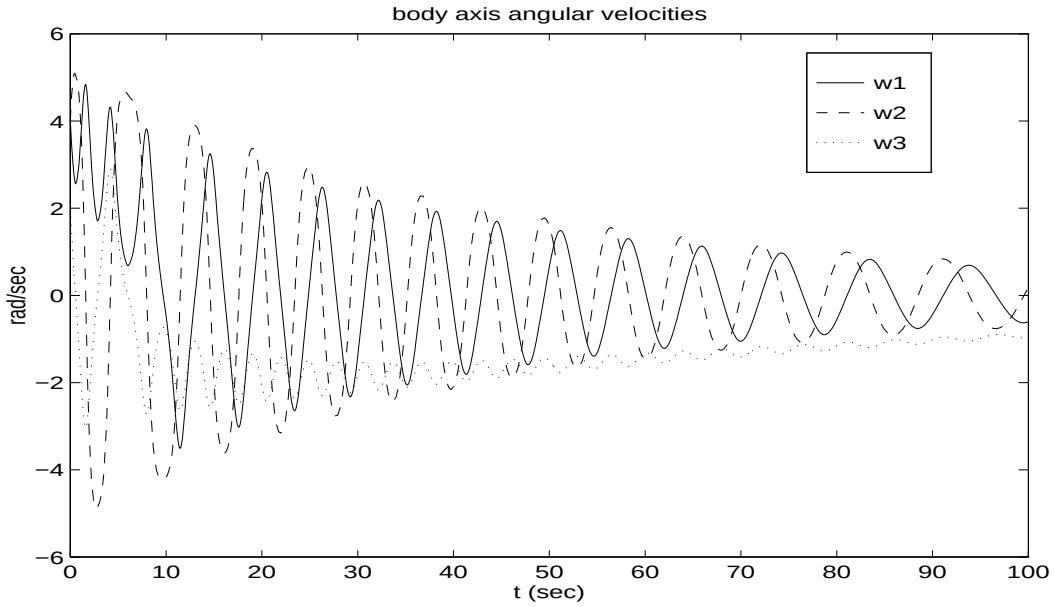

Figure 5.9 PA with $R = 10Q$: Control History

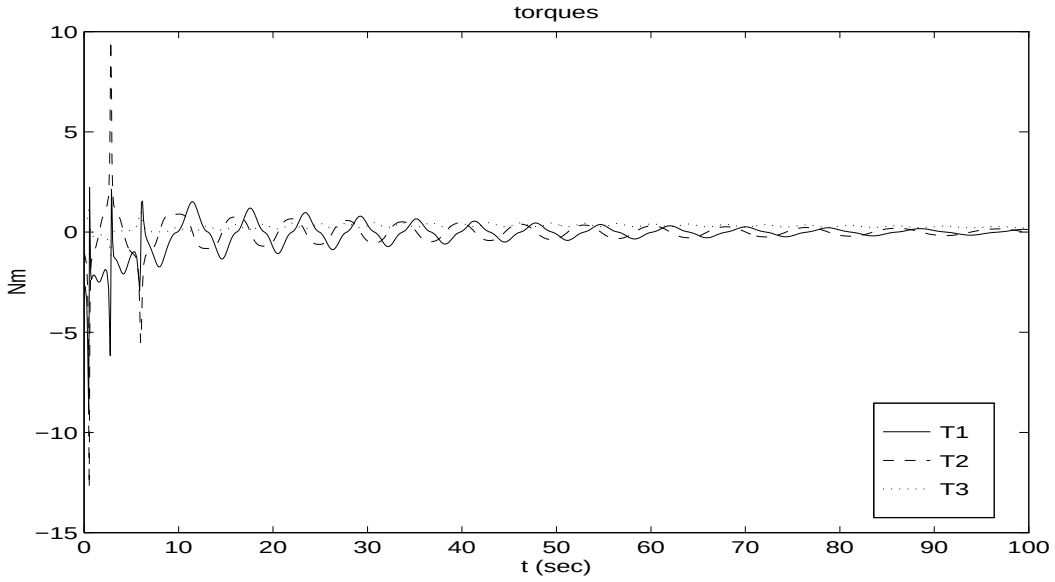

Figure 5.10 PA with $R = 10Q$: State Deviations

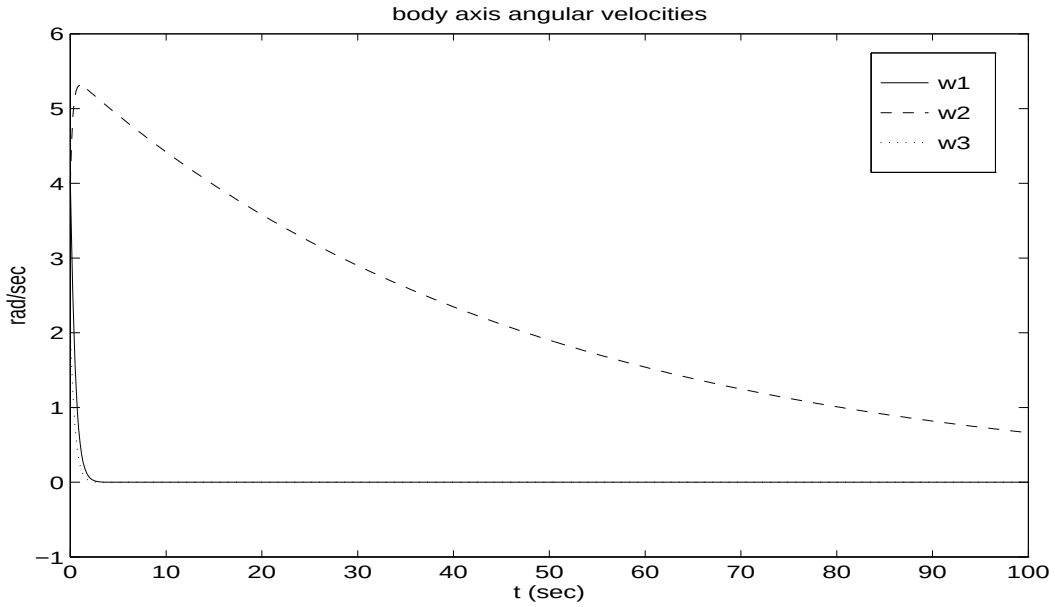

Figure 5.11 PC with $R = 10Q$: State Deviations

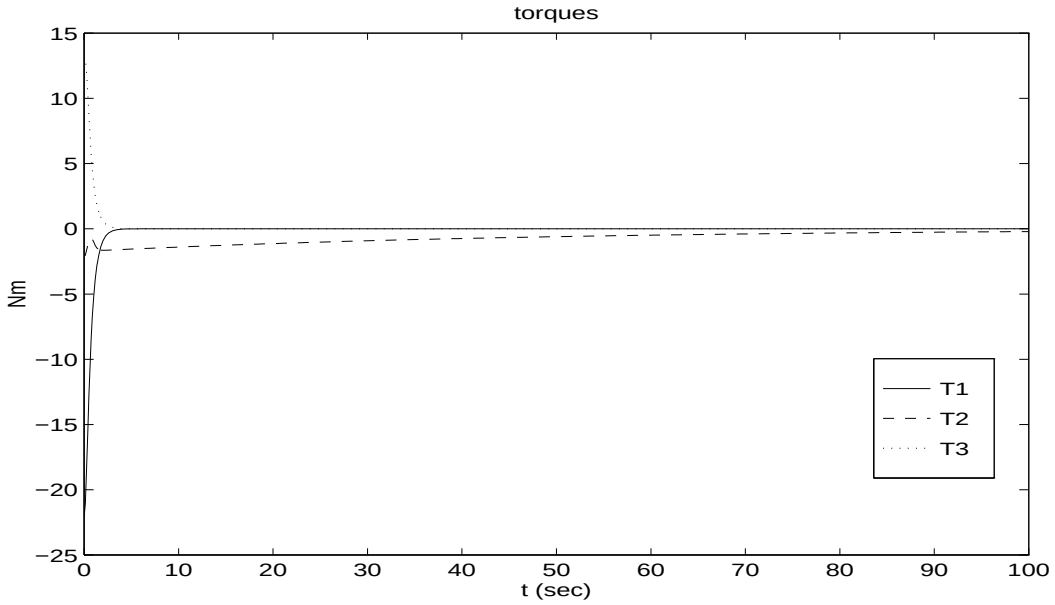

Figure 5.12 PC with $R = 10Q$: Control History

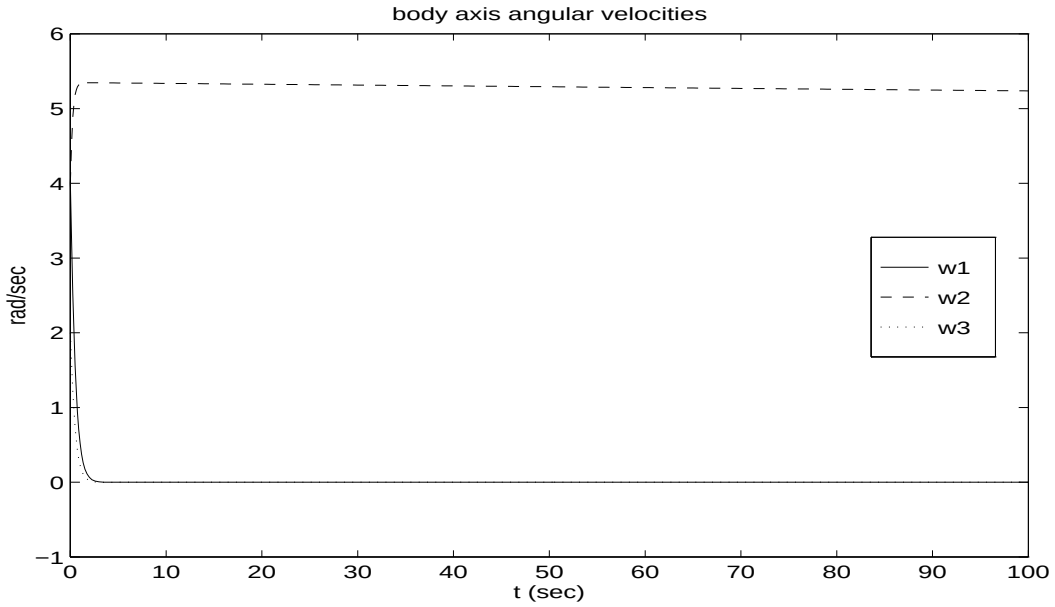

Figure 5.13 PC with $R = 10^5 Q$: State Deviations

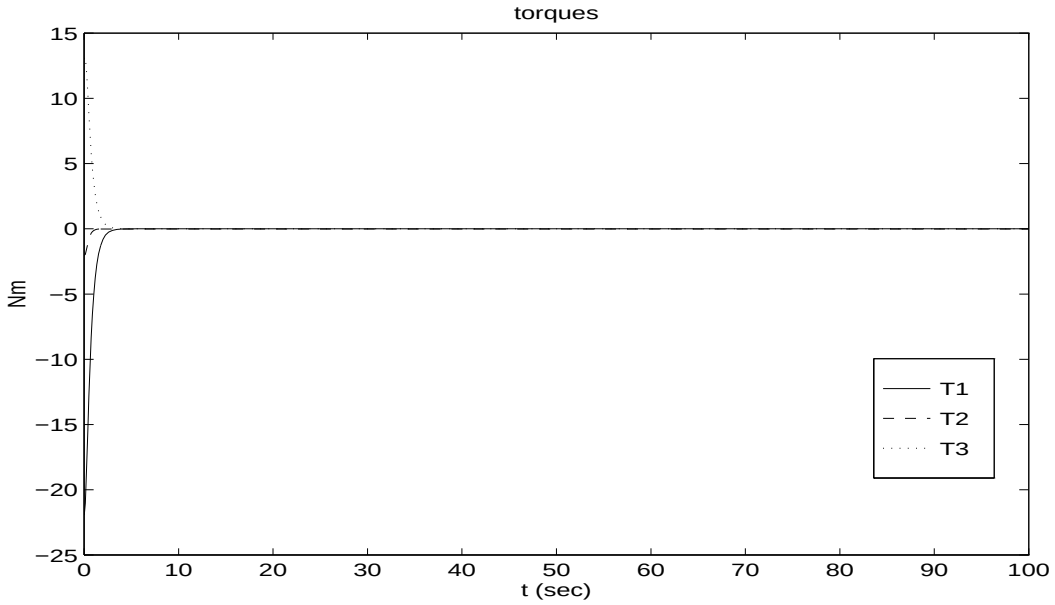

Figure 5.14 PC with $R = 10^5 Q$: Control History

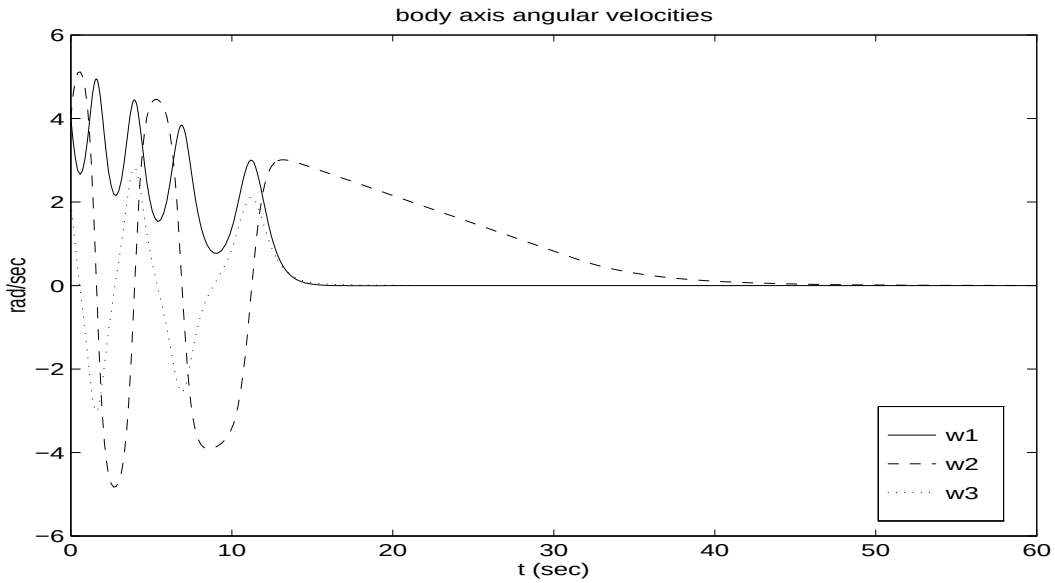

Figure 5.15 PC with $Q = 10R$, $\max|u_i| = 2$: State Deviations

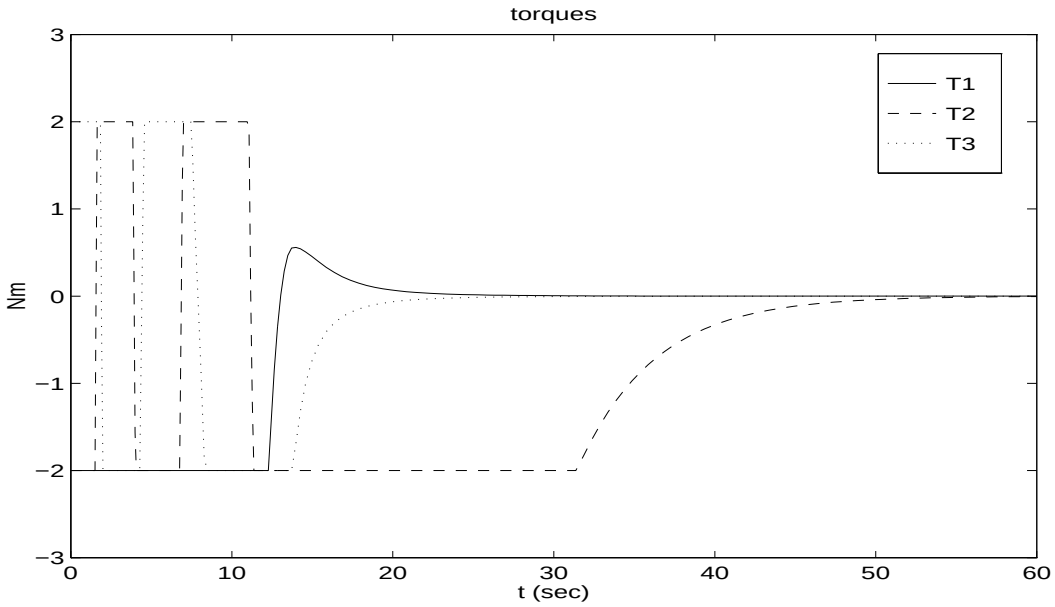

Figure 5.16 PC with $Q = 10R$, $\max|u_i| = 2$: Control History

The importance of optimality depends on each control problem examined. For our examples, each fixed sub-optimal parameterization resulted in an effective controller. If the control usage was too high for what would be implemented, the control could be limited. Implementing and testing a fixed SDC form was relatively quick, and many permutations of penalty matrices could be evaluated. If none of the controllers provided acceptable state and control histories, then at least the insight gained from the fixed form could help in choosing penalty matrices before solving the optimal problem.

VI. Satellite Control Results

This chapter will show the effectiveness of the numerical SDARE solutions using the dynamics of Chapter IV. The two examples presented here provide more practical examples than the illustrative problem of Chapter V.

6.1 State Regulation vs. Tracking

Each controller in the examples of this chapter can be viewed as either a regulator or tracker. For consistency, we will be regulating the angular rates and the first three quaternions. To do this most effectively, the fourth quaternion will be left completely unpenalized.

A quaternion vector $q = [0\ 0\ 0\ 1]^T$ implies that the body axes and inertial axes line up. If this is not the case, the coordinates that are measured will have to be transformed such that the controller sees the correct inertial frame. An example of this implementation can be seen in Figure 6.1.

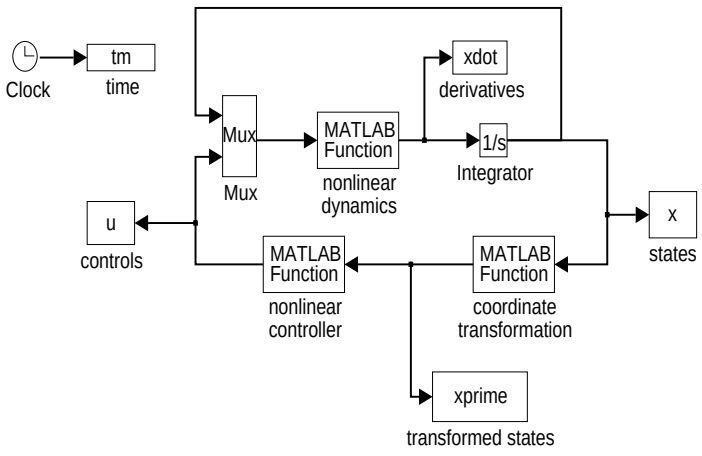

Figure 6.1 SIMULINK Regulator Model

In this same manner, the regulator can be made a tracker. To reorient the satellite, a different coordinate transformation merely needs to be done on the measured coordinates and then fed to the controller. To accomplish this, the quaternions would be transformed to a rotation matrix, then multiplied by the coordinate transformation. From this final rotation matrix the new quaternions can be calculated.

For systems whose states can all be driven to an equilibrium value, the coordinate transformation is not as complex. The tracking diagram for the pancreas model in Chapter VIII shows the more familiar method of subtracting reference values from the actual state values.

6.2 Despin of Satellite Using External Torques

This section will look at the following problem: given a perturbed rotating satellite, bring it to rest in a specific orientation.

6.2.1 SDC Parameterization. Combining the dynamics of Section 4.1 with the quaternion coordinates of Section 4.3, we will parameterize the system as

$$\begin{bmatrix} \dot{\omega} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -J^{-1}\omega^{\times}J & 0 \\ \mathcal{Q} & 0 \end{bmatrix} \begin{bmatrix} \omega \\ q \end{bmatrix} + \begin{bmatrix} J^{-1} \\ 0 \end{bmatrix} T \tag{6.1}$$

This parameterization was chosen strictly because it is a straightforward implementation of the dynamics as derived, and handles the general case with non-diagonal inertia matrices. The model retains all the cross coupling information in the A matrix as the parameterization C model did in the previous chapter. There is a slight difference in the magnitudes of the coefficients because of the different distribution of the inertia terms.

6.2.2 Initial Conditions and Weighting Functions. We will use the same inertia matrix as in Chapter V,

$$J = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 20 \end{bmatrix} \tag{6.2}$$

The perturbed satellite will also be rotating with initial angular velocities of

$$\omega_0 = \begin{bmatrix} 1 \\ 1 \\ 4 \end{bmatrix} \tag{6.3}$$

and have initial coordinates of

$$q_0 = \begin{bmatrix} -1.6145293678186 35e-02 \\ 4.399467145509192e-01 \\ 4.307255588328425e-01 \\ 7.878208621355699e-01 \end{bmatrix} \tag{6.4}$$

which corresponds to a 3-2-1 Euler axis rotation of $\phi = 70^o, \theta = 45^o$, and $\psi = 30^o$.

6.2.3 Simulation Results. For the first simulation we will use the following constant penalty matrices:

$$Q = \text{diag}\left(\begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 & 0 \end{bmatrix}\right) \tag{6.5}$$

$$R = I_{3\times 3} \tag{6.6}$$

The results can be seen in Figures 6.2 through 6.5. The controller is quite effective in stabilizing the satellite. An interesting feature of the NQR controller is the discontinuous control usage. Notice in Figure 6.4 that the sharpest change in control occurs at about 1 and 3.5 seconds. This

corresponds to where the unweighted quaternion q_4 is zero, which is also where ϕ takes its largest value.

For comparison, we will do a second simulation using the following constant penalty matrices:

$$Q = \text{diag}\left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}\right) \tag{6.7}$$

$$R = 5I_{3\times 3} \tag{6.8}$$

The results of the second weighting can be seen in Figures 6.6 through 6.9. This controller behaves as we would expect it to. With the reduced control usage, it takes almost twice as long to stabilize. The behavior is more oscillatory since the satellite will undergo more rotations during the regulation process. What is interesting to note here is that the unweighted quaternion q_4 went to -1, unlike $+1$ in the previous simulation, yet the final orientation of the satellite is the same. This confirms that we can pose a "stabilizable" problem, even though we had a nonstabilizable Hamiltonian matrix.

6.3 Reorientation of Internally Stabilized Satellite

For this example we consider a tracking problem. Given a satellite at rest in one initial position, bring the satellite to a new rest orientation with minimal angular velocities during the transition. The final orientation will be a body axis frame which lines up with the inertial reference frame; in other words, the inertial frame for the controller must be defined as that desired final state, as discussed in the beginning of this chapter.

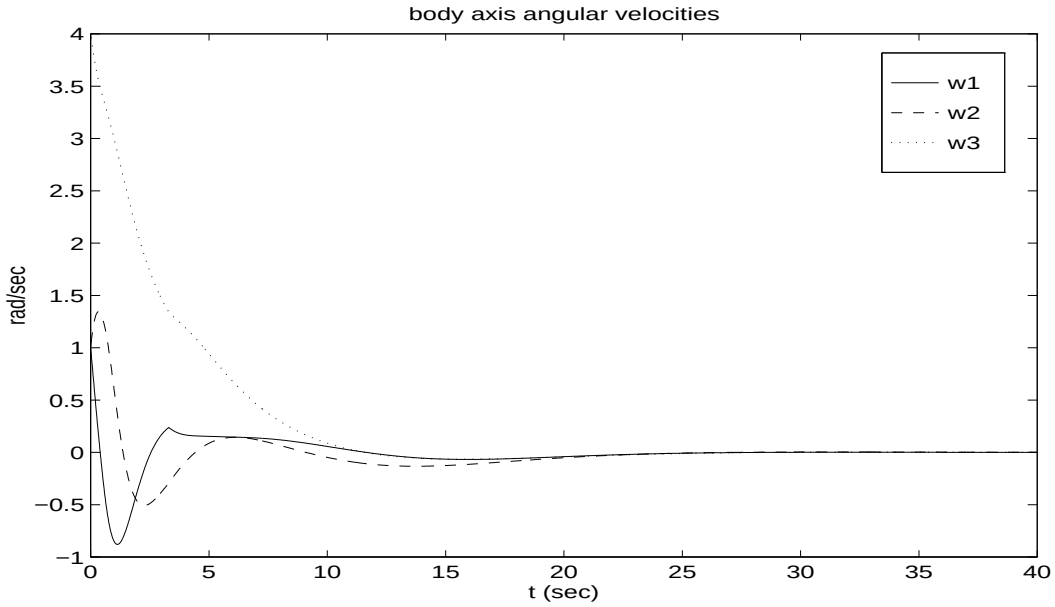

Figure 6.2 External Control, Heavy State Penalty: ω

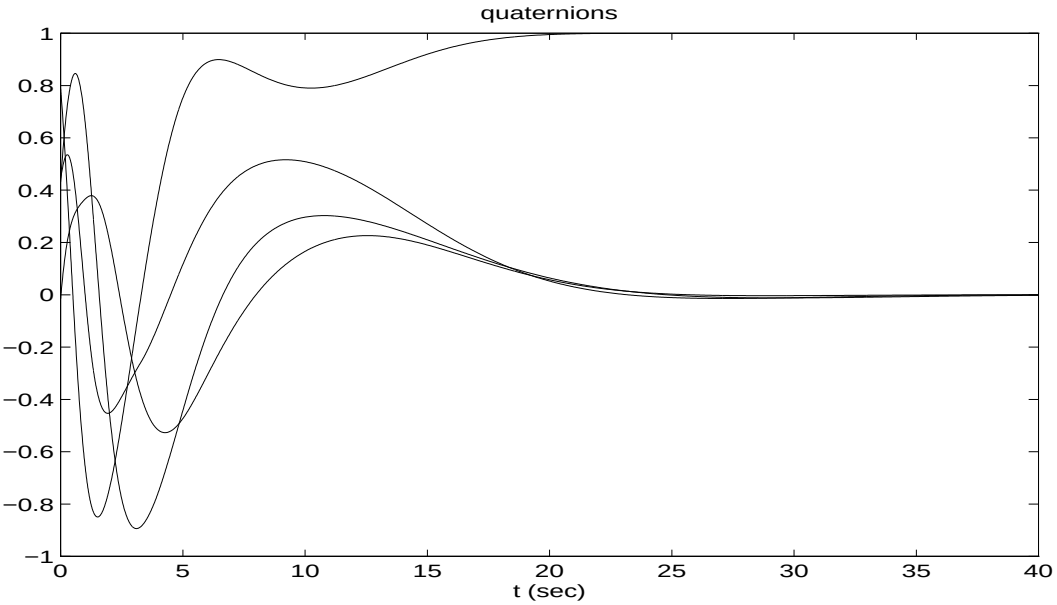

Figure 6.3 External Control, Heavy State Penalty: q

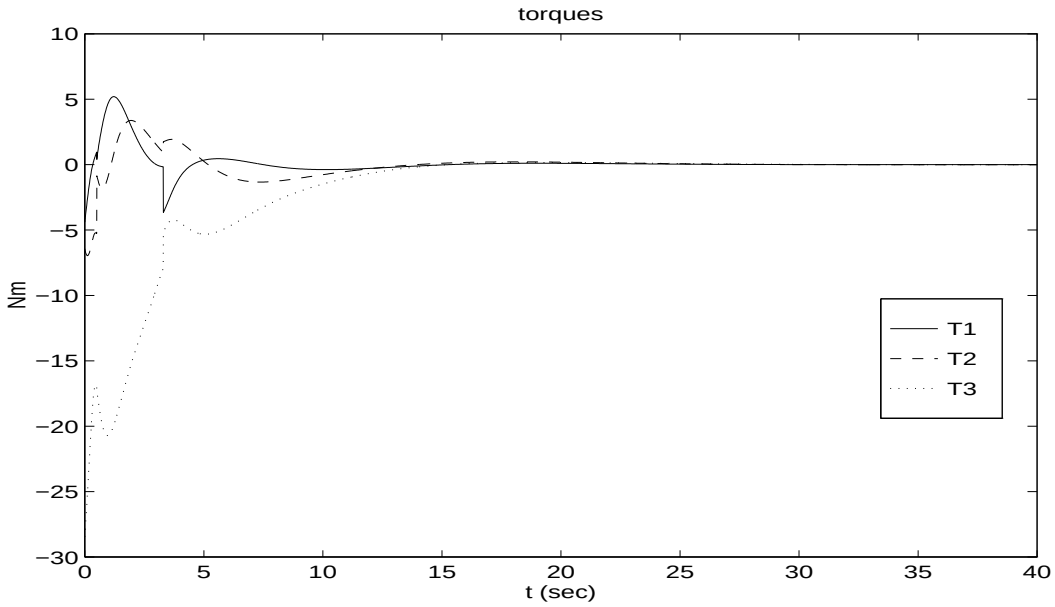

Figure 6.4 External Control, Heavy State Penalty: u

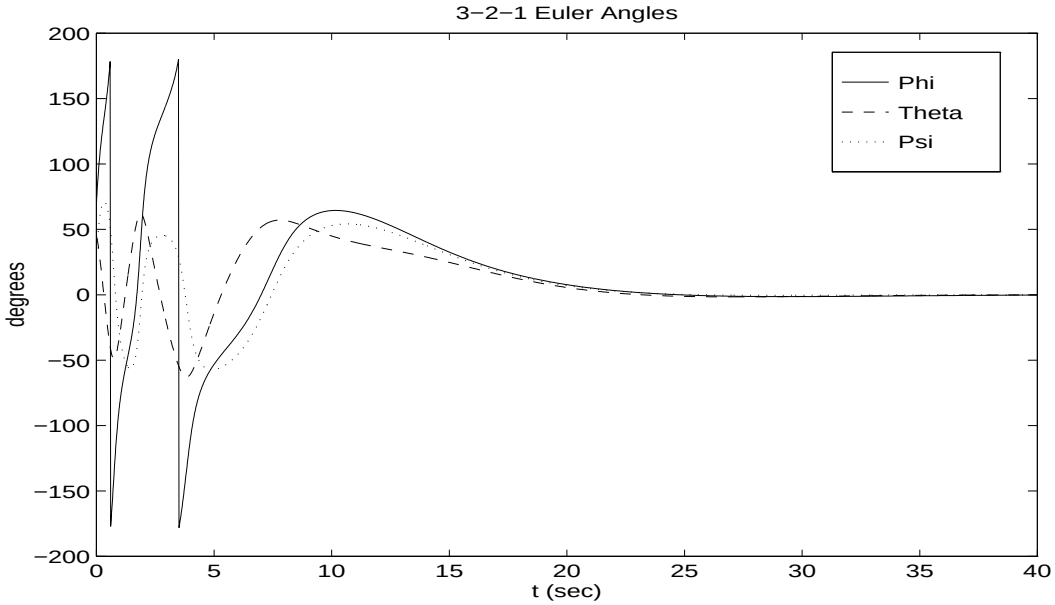

Figure 6.5 External Control, Heavy State Penalty: ϕ, θ, and ψ

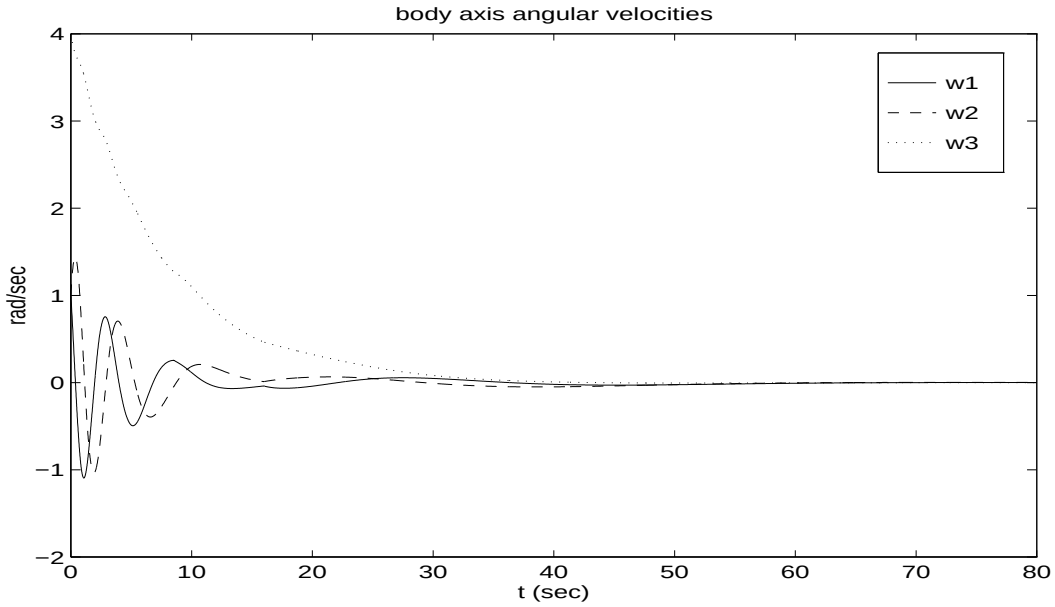

Figure 6.6 External Control, Heavy Control Penalty: ω

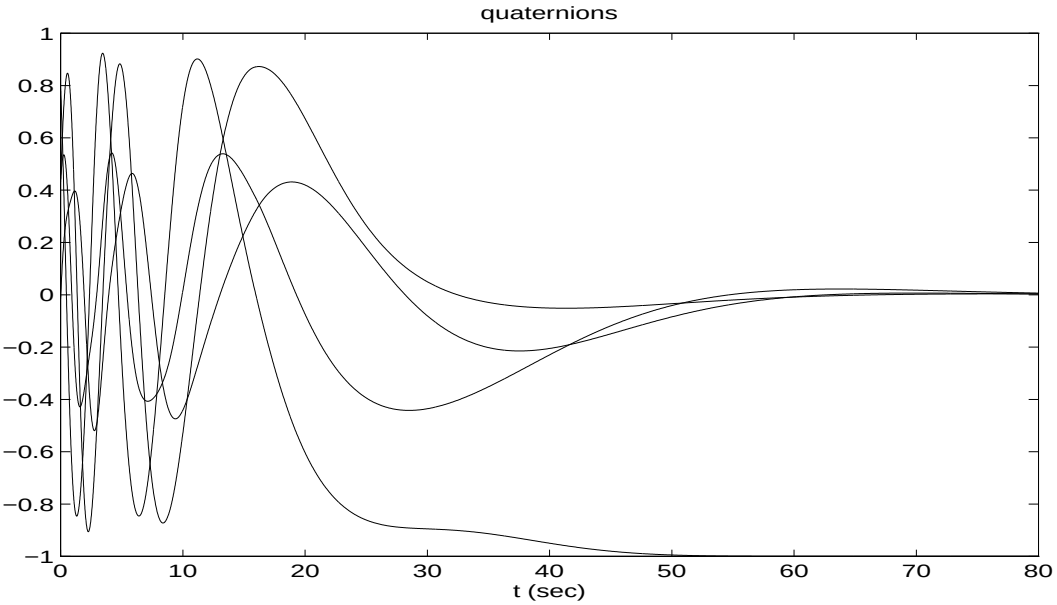

Figure 6.7 External Control, Heavy Control Penalty: q

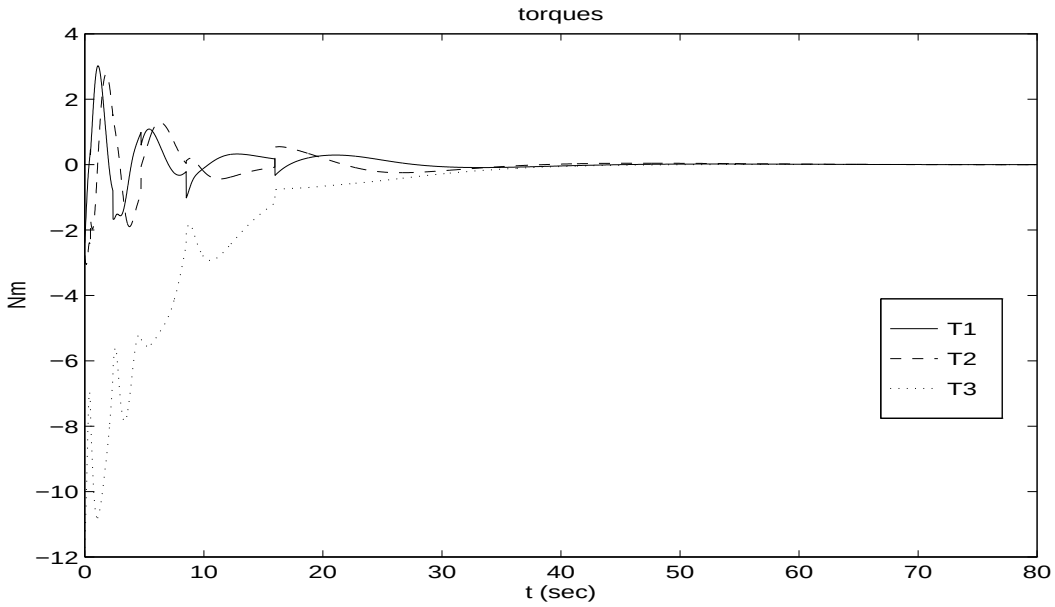

Figure 6.8 External Control, Heavy Control Penalty: u

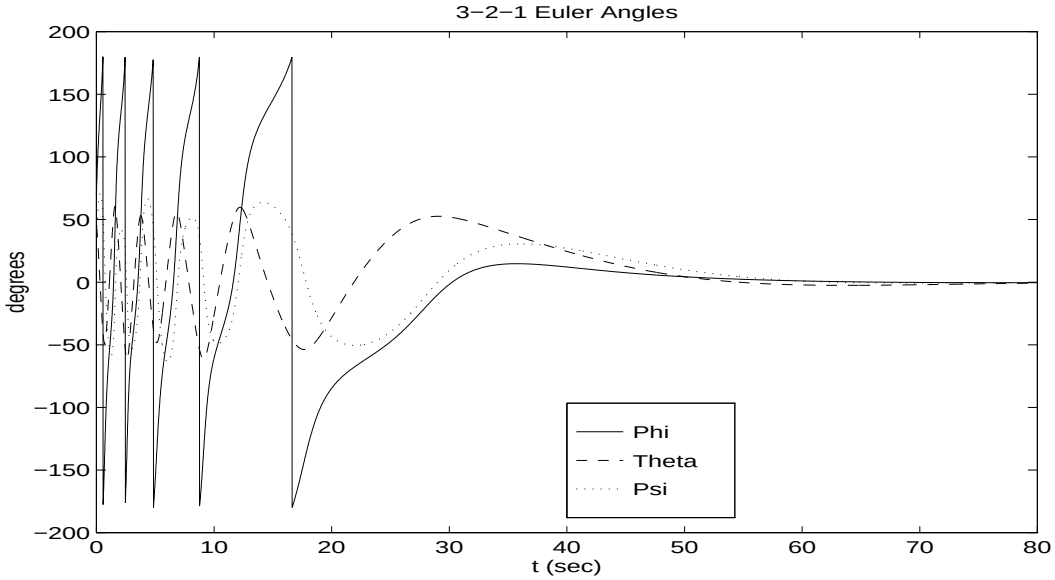

Figure 6.9 External Control, Heavy Control Penalty: ϕ, θ, and ψ

6.3.1 SDC Parameterization. Using the dynamics of Section 4.2 and differentiating Eqn. (4.5), we can form the following SDC factorization

$$\begin{bmatrix} \dot{\mu} \\ \dot{x} \\ \dot{\omega} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & x^{\times} & 0 \\ -J^{-1}x^{\times}J^{-1}A & J^{-1}x^{\times}J^{-1} & 0 & 0 \\ 0 & 0 & \mathcal{Q} & 0 \end{bmatrix} \begin{bmatrix} \mu \\ x \\ \omega \\ q \end{bmatrix} + \begin{bmatrix} I \\ 0 \\ J^{-1}A \\ 0 \end{bmatrix} u \tag{6.9}$$

remembering that μ is each rotor's angular momentum expressed in body axes, and x is the scaled angular momentum of the entire satellite expressed in body coordinates. For our examples, μ and x are three dimensional vectors.

This factorization is rather large and has some advantages and disadvantages associated with it. With this form, both angular rates and positions can be regulated and penalized accordingly. This SDC form does, however, contain redundant information; i.e., specifying the quaternions will uniquely define the body axis momentum. This parameterization expresses the dynamics more efficiently using the state x.

It is important to notice that the state μ is not desired to be regulated to zero, since it is the rotors' angular rates that will keep the satellite stationary. Therefore, μ should be left unpenalized and thus undetectable.

Satellite attitude is best regulated through q, and therefore x should also be unweighted. The reason x is not a good choice for attitude control, even though it is an inertially constant vector, involves the constraint $\|x\| = 1$. Like q, we could choose to make all but one coordinate go to zero. This is easily done; however, unlike q the position of the satellite is not uniquely defined in two senses. The x_n which is unweighted can go to either $+1$ or -1, which are indeed different answers. Also, even if x did achieve the desired final values, this only guarantees that the body vector and inertial vector are colinear. There is no unique rotation of the satellite about that axis.

As an aside, we will examine a second parameterization to illustrate the importance of forming a numerically well conditioned problem. Rather than using the simplified version of Eqn. (4.3) as above, the $\dot{x}$ equation could have been used directly as in the following SDC form

$$\begin{bmatrix} \dot{\mu} \\ \dot{x} \\ \dot{\omega} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -x^{\times}J^{-1}A & x^{\times}J^{-1} & 0 & 0 \\ -J^{-1}x^{\times}J^{-1}A & J^{-1}x^{\times}J^{-1} & 0 & 0 \\ 0 & 0 & \mathcal{Q} & 0 \end{bmatrix} \begin{bmatrix} \mu \\ x \\ \omega \\ q \end{bmatrix} + \begin{bmatrix} I \\ 0 \\ J^{-1}A \\ 0 \end{bmatrix} u \tag{6.10}$$

This is an example of the problem discussed in section 3.1.3. While it appears as though it should be a valid factorization, numerically the Hamiltonian is badly scaled. The similarities between the second and third submatrix rows of the A matrix causes numerical problems. Here the A matrix only has a block rank of 2, as compared to a block rank of 3 in the first parameterization.

6.3.2 Initial Conditions and Weighting Functions. Here μ and x will be left unweighted for the reasons already stated. Attitude control will be accomplished by weighting the first three elements of q.

We will use a scaled inertia matrix of

$$J = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.75 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.11}$$

Using three rotors with directions given by

$$A = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \tag{6.12}$$

and given initial rotor momentum of

$$\mu_0 = \begin{bmatrix} 0.2 \\ 0.6 \\ 0.4 \end{bmatrix} \tag{6.13}$$

we calculate the normalized initial satellite momentum vector to be

$$x_0 = \begin{bmatrix} 4.2426e-01 \\ 5.6569e-01 \\ 7.0711e-01 \end{bmatrix} \tag{6.14}$$

where $x = A\mu$ so the satellite starts from rest.

We will use the same initial coordinates from our externally controlled satellite of

$$q_0 = \begin{bmatrix} -1.614529367818635e-02 \\ 4.399467145509192e-01 \\ 4.307255588328425e-01 \\ 7.878208621355699e-01 \end{bmatrix} \tag{6.15}$$

corresponding to a 3-2-1 Euler axis rotation of $\phi = 70^o, \theta = 45^o$, and $\psi = 30^o$.

The penalty matrices for the first simulation will be:

$$Q = \text{diag}\left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 10 & 10 & 10 & 10 & 10 & 10 & 0 \end{bmatrix}\right) \tag{6.16}$$

$$R = I_{3\times 3} \tag{6.17}$$

while the second simulation will have increased relative control weighting. The penalty matrices for this set up will be:

$$Q = \text{diag}\left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}\right) \tag{6.18}$$

$$R = 50 I_{3\times 3} \tag{6.19}$$

6.3.3 Simulation Results. Results for the first weighting set are given in Figures 6.10 through 6.15. Notice that this controller is very effective in reorienting the satellite. Of all the examples presented in this thesis, this is perhaps the most promising in real application. The controller induces a quick increase in body axis angular velocities, and then exponentially decays them back to their rest position. Control usage is very smooth. Also remember that the dynamics are scaled, so that the time to settle does not represent real seconds, but rather scaled time units.

For our second example we will see how a larger control penalty impacts the problem. The results are seen in Figures 6.16 through 6.21. This controller still provides excellent stability properties. Now, however, instead of the sharp rise and exponential decay in angular velocities, a sinusoidal pattern is superimposed over the growth and decay. Additionally, the settling time has also increased. Further penalizing control usage only increases this sinusoidal nature, with a continuing increase in time to settle.

6.4 Summary

This chapter shows that using a state dependent algebraic Riccati equation results in effective control usage which can be used to both stabilize and reorient a satellite. The first system demonstrated the ability of NQR to command external torques to bring a satellite to rest. Since control usage is limited by the fuel reserves of the satellite, this is not a very practical example. This example was meant to help validate NQR as a control method.

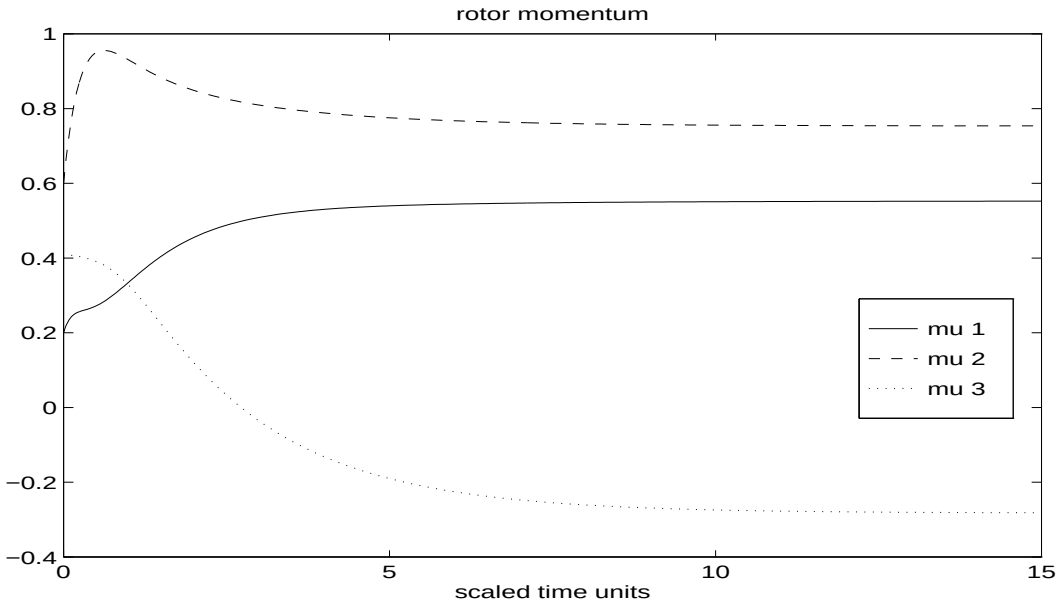

Figure 6.10 Internal Control, Heavy State Penalty: μ

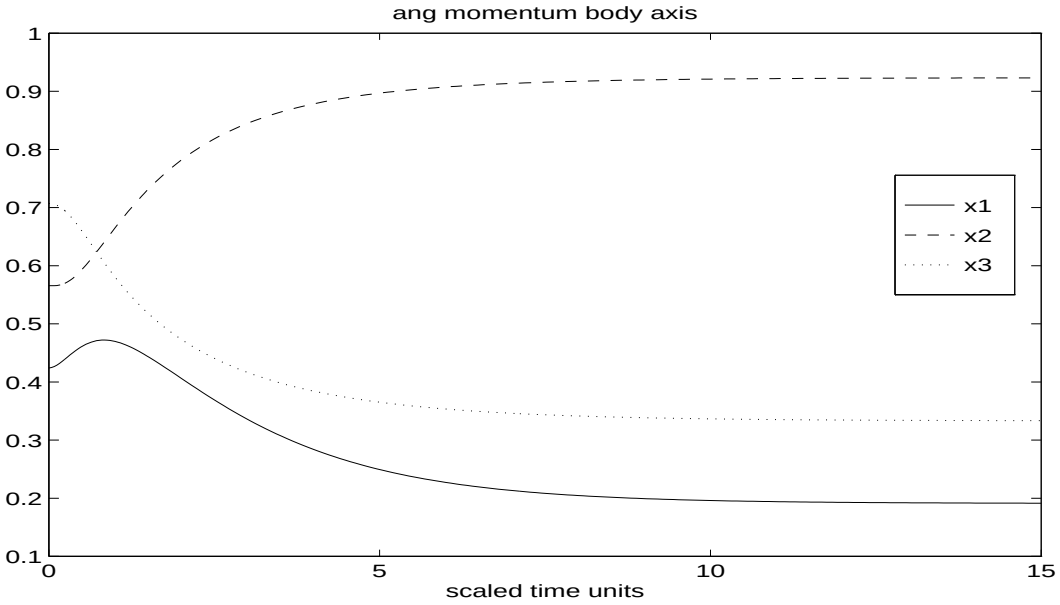

Figure 6.11 Internal Control, Heavy State Penalty: x

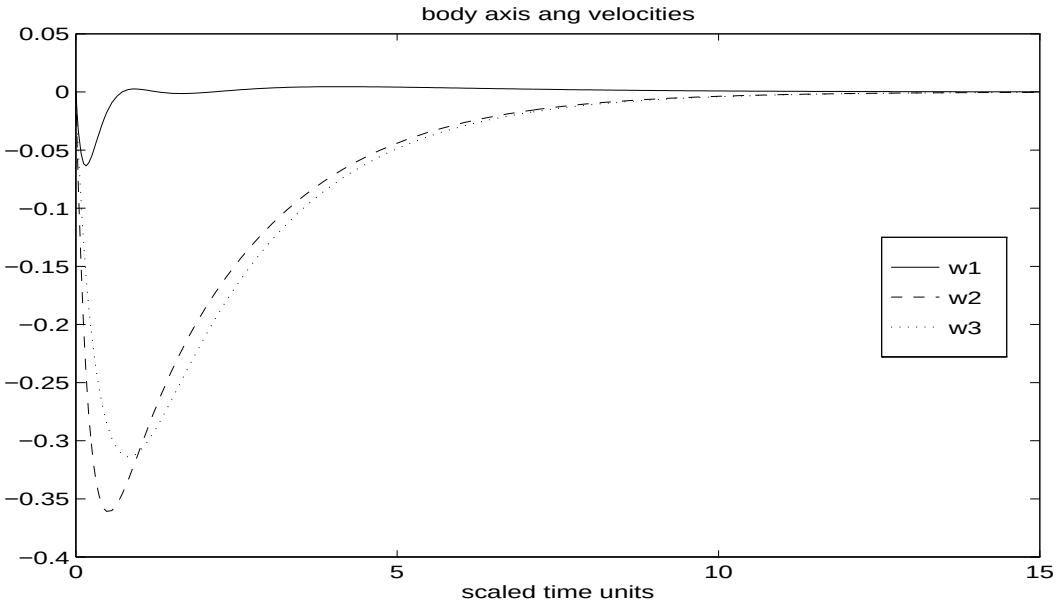

Figure 6.12 Internal Control, Heavy State Penalty: ω

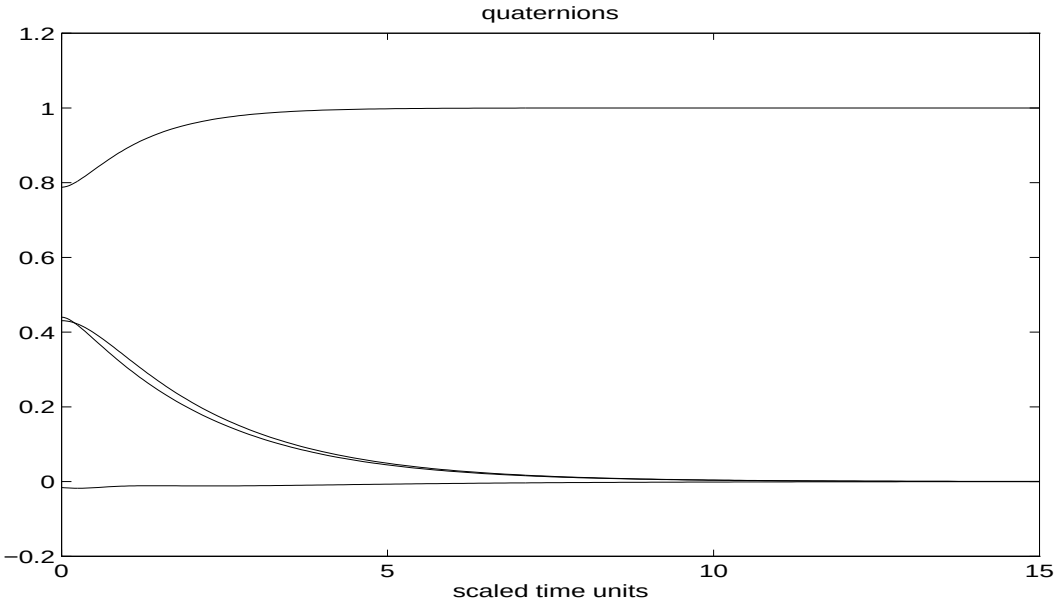

Figure 6.13 Internal Control, Heavy State Penalty: q

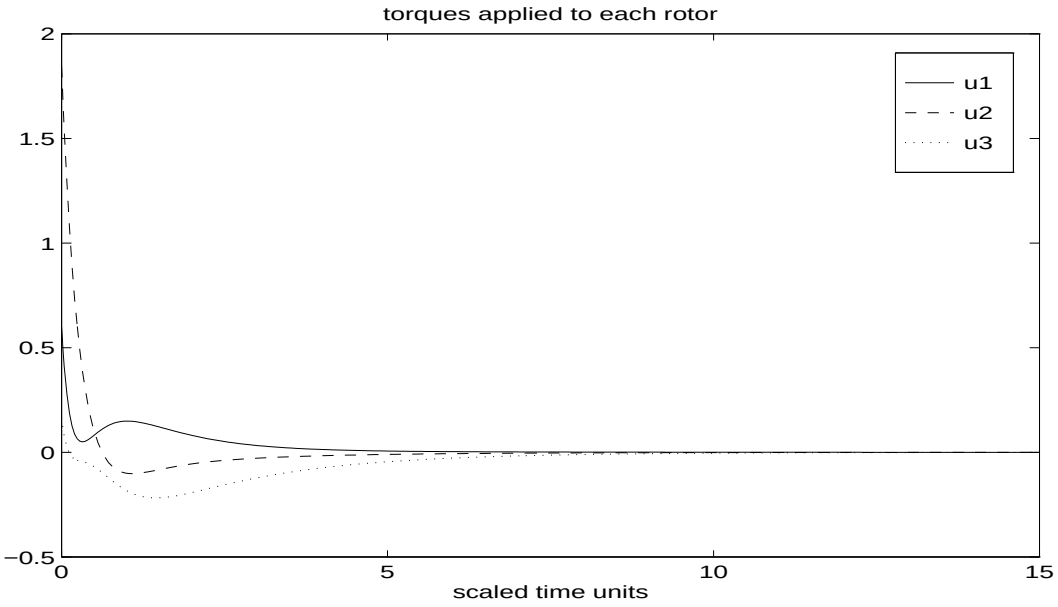

Figure 6.14 Internal Control, Heavy State Penalty: u

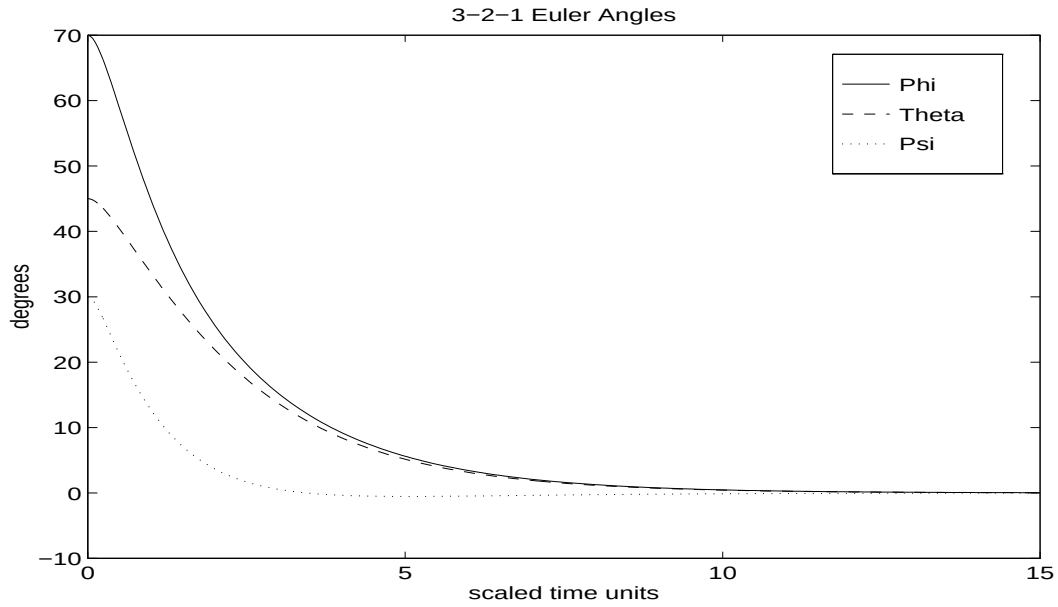

Figure 6.15 Internal Control, Heavy State Penalty: ϕ, θ, and ψ

The internally controlled satellite example provides a more practical application for NQR. The satellite can be efficiently reoriented with a relatively simple control law. Full state feedback does not provide a limiting factor here, since measuring accelerations about all three axes is easily implementable. These results are very encouraging. If a controller was implemented at an appropriately high enough frequency, this NQR method would provide one more alternative for controlling internally stabilized satellites.

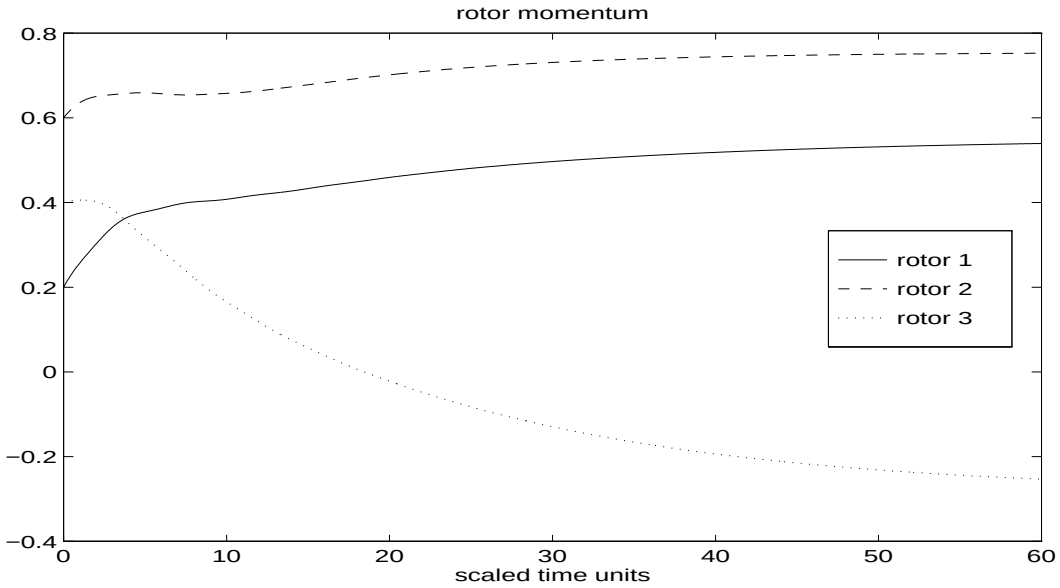

Figure 6.16 Internal Control, Heavy Control Penalty: μ

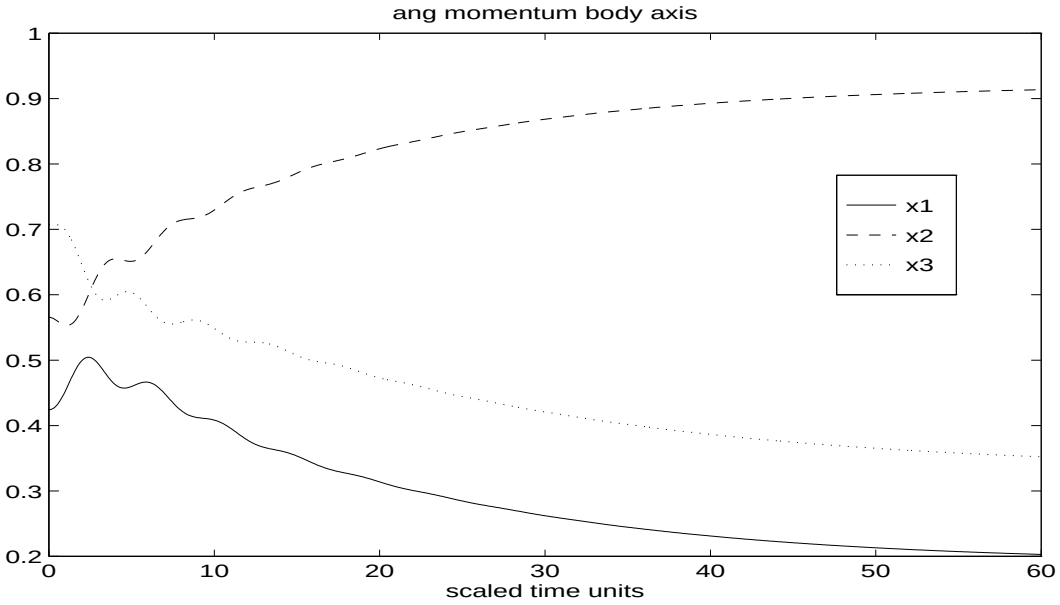

Figure 6.17 Internal Control, Heavy Control Penalty: x

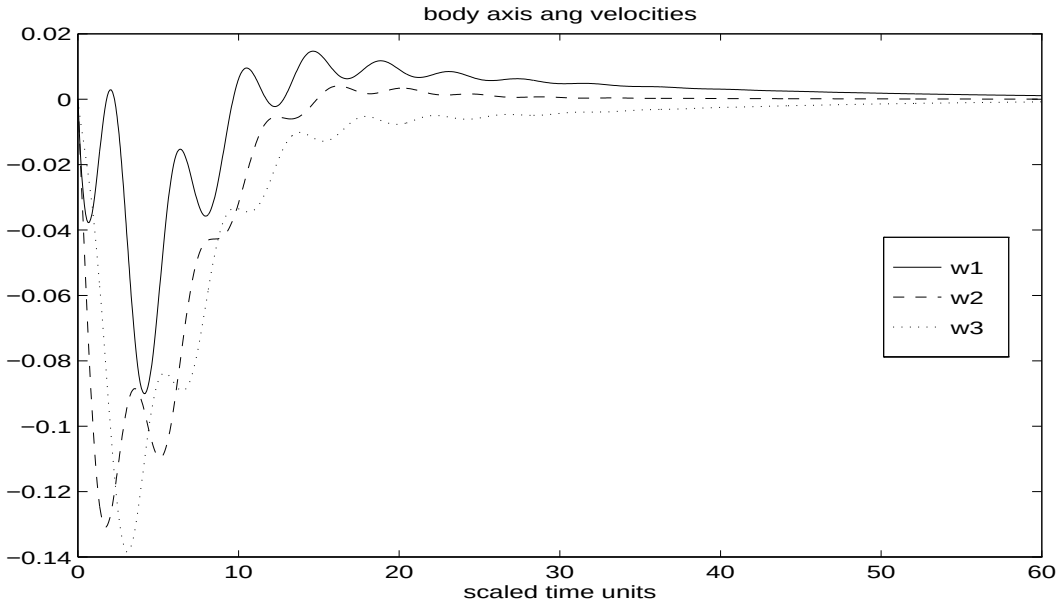

Figure 6.18 Internal Control, Heavy Control Penalty: ω

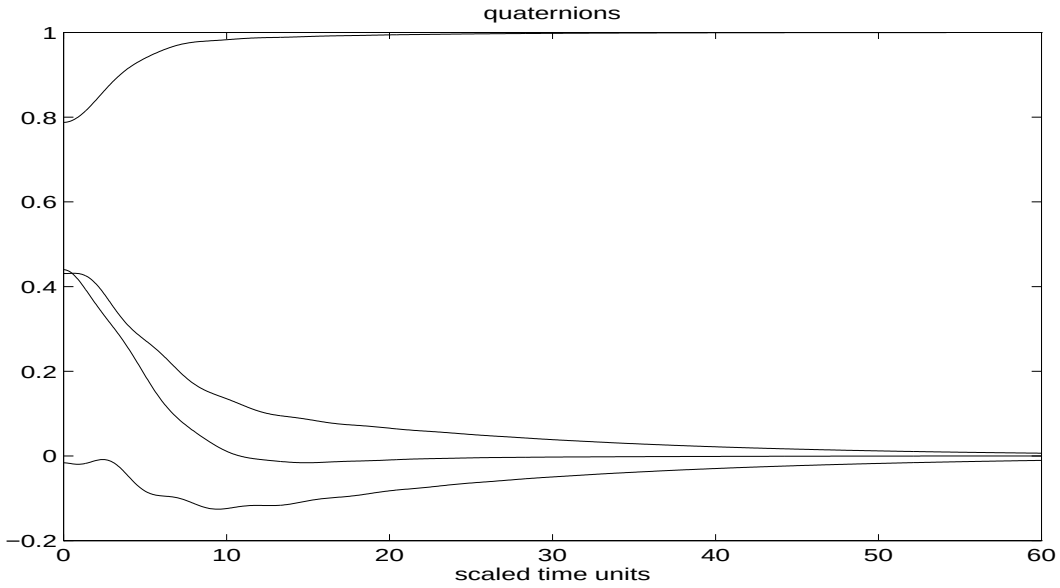

Figure 6.19 Internal Control, Heavy Control Penalty: q

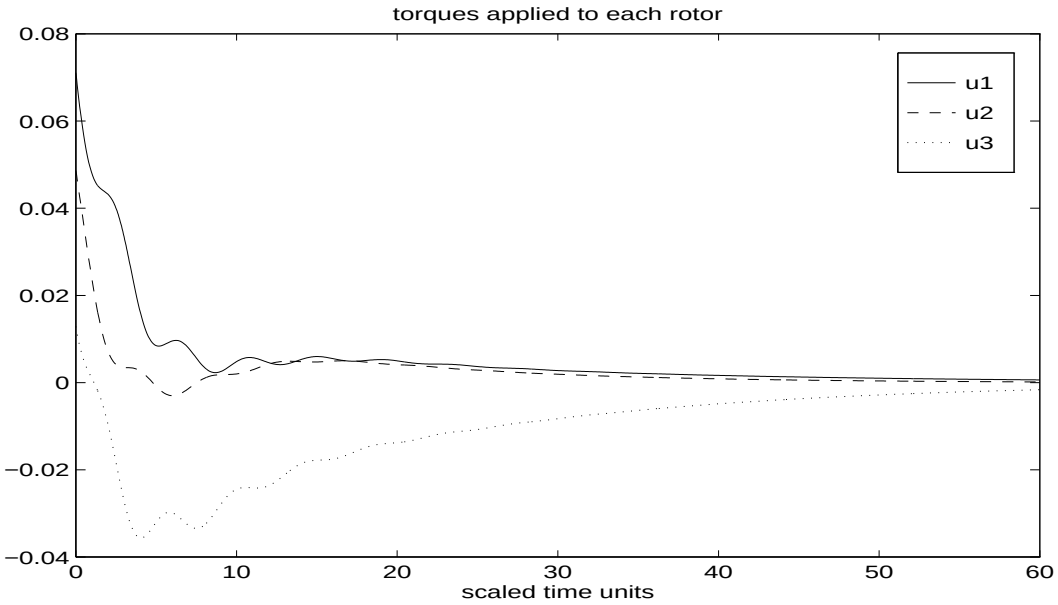

Figure 6.20 Internal Control, Heavy Control Penalty: u

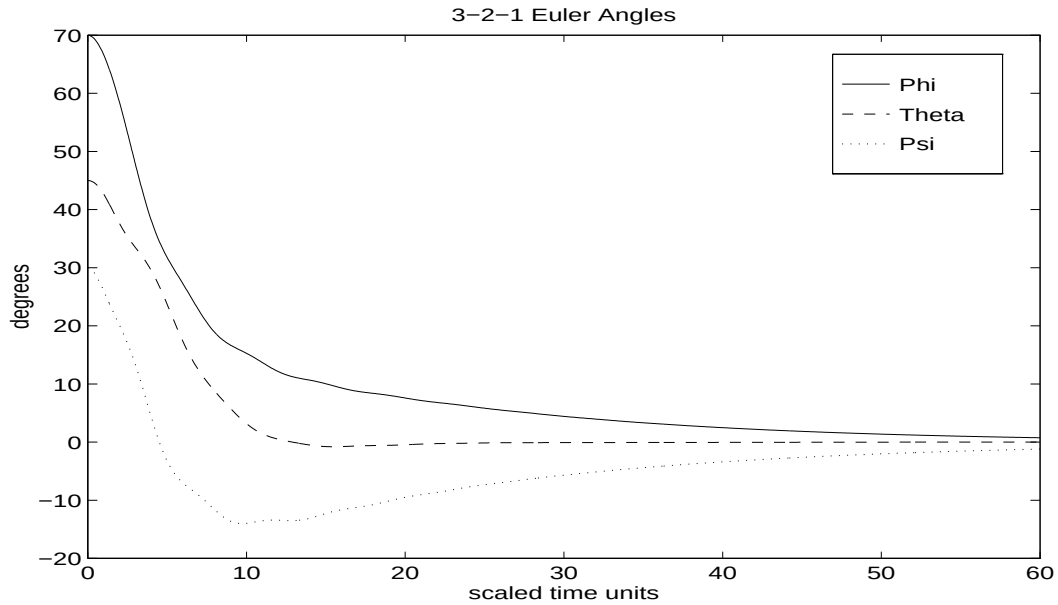

Figure 6.21 Internal Control, Heavy Control Penalty: ϕ, θ, and ψ

VII. Artificial Pancreas Model

This chapter covers the modeling of human glucose and insulin dynamics as developed by Hodel and Naylor [NHS95] of Auburn University. The model was developed in SIMULINK and uses modules written in C. This model, the Advanced Endocrine Management of Glucose (AEMG) model, is divided into four compartments simulating the liver, pancreas, and blood and tissue chemistry. For further information, see the individual papers by Hodel [Hod94] and Naylor [Nay94]. The dynamics presented here consolidate the information of those four modules. No claims are made in this thesis as to the accuracy or validity of these equations, since verification is beyond the scope of this thesis. We are merely concerned with the following problem: assuming these dynamics, what kind of performance does a nonlinear quadratic regulator achieve?

The complexity of the AEMG model is an attempt to present a better model of glucose and insulin dynamics. The original intent of this thesis was to investigate linear controllers for this model. Linear controllers examined in earlier papers and in Hodel's paper were based on simpler models and were determined to be inadequate. When this author tried to linearize the larger AEMG model dynamics, it was determined that linearizing about equilibrium was valid for only insignificant perturbations. A comparison of the linear and nonlinear dynamics in response to an external glucose disturbance is shown in Figure 7.1. The linear response is not large enough to warrant any control, whereas the 'real' response is significant and requires insulin to counter the effect. This prompted the investigation into nonlinear control techniques.

7.1 Nonlinear Dynamics

The current AEMG model contains nine states given in Table 7.1, and it is highly nonlinear. The only changes to the dynamics presented here were altering the high and low limits of some trigger functions and the addition of a constant term in the glucose state equation. This was done so the equations would have a steady state value equal to average normal levels in humans. The

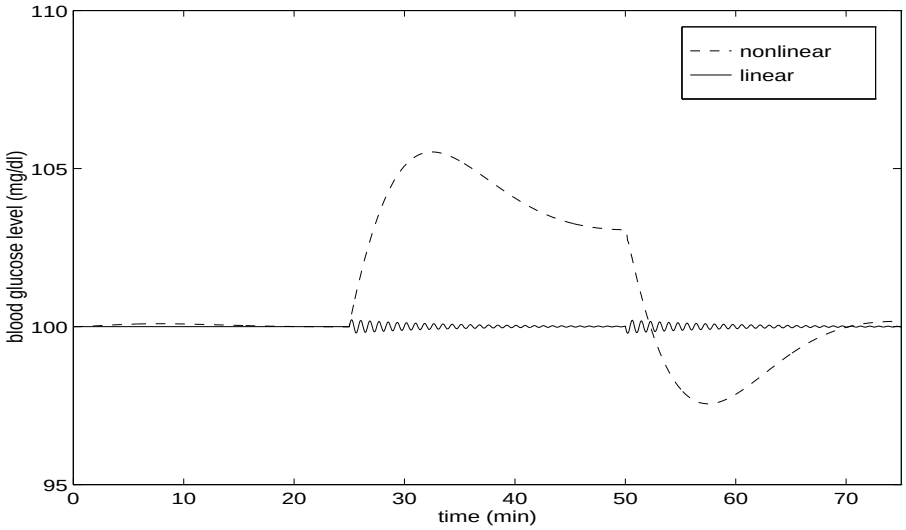

Figure 7.1 Linear vs. Nonlinear Response

code received by the author was designed to settle with a glucose blood level at a slightly higher value. The dynamics behave in the same way, but with these changes the steady state of the system matched the average basal (equilibrium) level of humans as given in [Hod94].

Table 7.1 AEMG State Variables

x_{cl}	cortisol level
x_{el}	epinephrine level
x_{ggl}	glucagon level
x_{gl}	glucose level
x_{ghl}	human growth hormone level
x_{il}	insulin level
x_{gngl}	gluconeogenesis level
x_{gs}	glucose stores
x_{is}	insulin stores

The equations are presented in the form they are to help minimize the effort to correlate them with the computer code of the AEMG model and previous papers. Therefore, coefficients are not always reduced to a single number or constants substituted with their values. Explanation of the dynamics and the terms found in the equations will follow the model description.

The dynamical model is:

$$\dot{x}_{cl} = -c_{cl}x_{cl} + c_{cl}b_{cl}\frac{1+\text{trigh}(x_{gl},90,60)}{1+\text{trigh}(b_{gl},90,60)} \tag{7.1}$$

$$\dot{x}_{el} = -c_{el}x_{el} + c_{el}b_{el}\frac{\text{trigh}(x_{gl},1.2b_{gl},0.8b_{gl})}{\text{trigh}(b_{gl},1.2b_{gl},0.8b_{gl})} \tag{7.2}$$

$$\dot{x}_{ggl} = -c_{ggl}x_{ggl} + 0.1(1.0-\beta)c_{ggl}b_{ggl} + \frac{0.4x_{gs}\text{trigh}(x_{gl},140\tilde{\alpha},60\tilde{\alpha})}{1000\cdot vol} \tag{7.3}$$

$$\begin{aligned}\dot{x}_{gl} = \Bigg[&24\cdot 60\cdot 2mass\cdot x_{gngl} - 0.75\cdot 2mass\frac{\text{trigh}(x_{gl},60,150)}{\text{trigh}(b_{gl},60,150)}\\ &+0.91\cdot 2mass\frac{\text{trigh}(\frac{x_{el}}{b_{el}},0.9,5.0)+\text{trigh}(\frac{x_{ggl}}{b_{ggl}},0.9,5.0)-\frac{x_{gl}}{b_{gl}}\text{trigh}(\frac{x_{il}}{b_{il}},0.25,5.0)}{2\text{trigh}(1.0,0.9,5.0)-\text{trigh}(1.0,0.25,5.0)}\\ &-0.5mass\frac{x_{gl}x_{il}}{b_{gl}b_{il}} - 47.999232 + w_{\text{glucose}}\Bigg]\frac{1}{10\cdot vol}\end{aligned} \tag{7.4}$$

$$\dot{x}_{ghl} = -c_{ghl}x_{ghl} + c_{ghl}b_{ghl}\frac{1+\text{trigh}(x_{gl},90,60)}{1+\text{trigh}(b_{gl},90,60)} \tag{7.5}$$

$$\dot{x}_{il} = -c_{il}x_{il} + \frac{0.4x_{is}\text{trigh}(x_{gl},60,140)}{1000\cdot vol} + \frac{u_{\text{insulin}}}{vol} \tag{7.6}$$

$$\begin{aligned}\dot{x}_{gngl} = &-c_{gngl}x_{gngl} + \frac{c_{gngl}b_{gngl}}{2\text{trigh}(1.0,1.0,5.0)+2\text{trigh}(1.0,0.9,5.0)-\text{trigh}(1.0,0.5,5.0)}\cdot\\ &\Bigg[\text{trigh}(\frac{x_{cl}}{b_{cl}},1.0,5.0)+\text{trigh}(\frac{x_{el}}{b_{el}},0.9,5.0)+\text{trigh}(\frac{x_{ggl}}{b_{ggl}},0.9,5.0)+\\ &\text{trigh}(\frac{x_{ghl}}{b_{ghl}},1.0,5.0)-\text{trigh}(\frac{x_{il}}{b_{il}},0.5,5.0)\Bigg]\end{aligned} \tag{7.7}$$

$$\dot{x}_{gs} = 0.2\alpha(m_{gs} - x_{gs}) - 0.4x_{gs}\text{trigh}(x_{gl}, 140\tilde{\alpha}, 60\tilde{\alpha})$$
$$-0.1(1.0 - \beta)c_{ggl}b_{ggl} \cdot 1000vol \tag{7.8}$$

$$\dot{x}_{is} = 0.2\beta(m_{is} - x_{is}) - 0.4x_{is}\text{trigh}(x_{gl}, 60, 140) \tag{7.9}$$

Control is done by the injection of insulin and is seen in the term u_{insulin} in the x_{il} state equation. Although the mathematical dynamics will allow negative control, it is not physically implementable and must be accounted for. This is done in the simulation by putting a limiter in the SIMULINK model. Also, to save computational time, the controller function can return a zero control command whenever x_{gl} is less than the desired level, without solving the SDARE. Disturbances to the model are modeled through the term w_{glucose} in the x_{gl} state equation, which models dietary intake of glucose. Negative disturbances could be used to model glucose uptake if needed.

The variables *mass* and *vol* represent the body mass and blood volume, respectively. Average values of 80kg body mass and 6 liters of blood were used for the simulation to match those used by [Hod94].

The constants c_{states} are related to the half lives (in minutes) of the respective hormones and given by

$$c = \frac{-\ln(0.5)}{\text{half life}} \tag{7.10}$$

The values used for the half lives are given in Table 7.2. These are only approximations, since there is not enough data yet to improve their accuracy. For example, the 7.5 values represent the median between possible values of 5 minutes to 10 minutes.

Table 7.2 Half Life Values in Minutes

state	value
cortisol	7.5
epinephrine	7.5
glucagon	7.5
glucose	7.5
human growth hormone	90.0
insulin	7.5
gluconeogenesis	30.0

The constants b_{states} are the basal levels of each state for an average healthy person and are found in Table 7.3. The values for the first seven states came from the literature [Hod94], while the last two were calculated from their equations in the AEMG code.

Table 7.3 Basal Levels

b_{cl}	11.0
b_{el}	92.0
b_{ggl}	120.0
b_{gl}	100.0
b_{ghl}	2.4
b_{il}	35.0
b_{gngl}	2.708e-4
b_{is}	3.4657e+5
b_{gs}	9.7041e+4

The variables α and β are used to simulate healthy versus diabetic people. For a healthy person, $\alpha = \beta = 1$. The diabetic model we wish to regulate will use $\alpha = \beta = 0.1$ which are the values used in [Nay94]. The variable $\tilde{\alpha}$ is a shift value given by

$$\tilde{\alpha} = 1.1 - 0.1\beta \tag{7.11}$$

which is used to shift the arguments of some trigger functions. This represents a change in effectiveness of one hormone responding to another.

The last variables m_{gs} and m_{is} are the maximum glucose stores and maximum insulin stores, respectively. These states represent a total amount stored in the tissue and bloodstream of a

human. They are defined to have the following values:

$$m_{gs} = 10 \cdot c_{ggl} \cdot b_{ggl} \cdot vol \cdot 1000 \tag{7.12}$$

$$m_{is} = 10 \cdot c_{il} \cdot b_{il} \cdot vol \cdot 1000 \tag{7.13}$$

7.2 Trigger function

The function trigh is used to scale the gains of terms it is found in. It has three arguments and is given by

$$\text{trigh}(x, l, h) \stackrel{d}{=} \frac{1}{2}\left[1 + \tanh\left(\frac{2x - l - h}{h - l}\right)\right] \tag{7.14}$$

The trigger function has a range of (0,1) for a domain over all x. A representative graph can be seen in Figure 7.2. For $h < l$ the function would start at 1 and decrease to 0 for larger x. Use

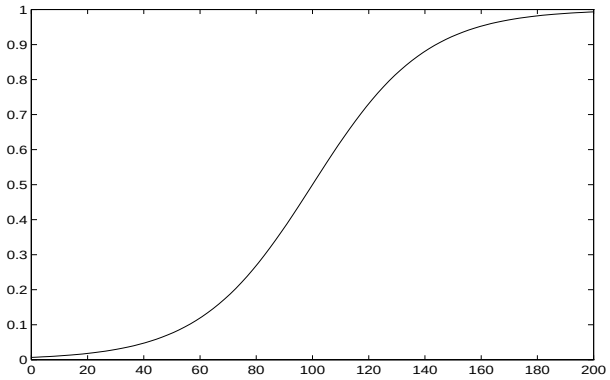

Figure 7.2 $y = \text{trigh}(x, 60, 140)$

of this function helps model one state's sensitivity to another by causing a cutoff and saturation effect. We will also make use of the derivative with respect to x, given by

$$\frac{d\text{trigh}(x, l, h)}{dx} = \frac{1}{(h - l)}\text{sech}^2\left(\frac{2x - l - h}{h - l}\right) \tag{7.15}$$

7.3 Control Objectives and Concerns

The purpose of examining automatic control is to eliminate the dangers diabetics face with elevated glucose levels. Type I diabetes, or insulin-dependent diabetes mellitus, requires daily injection of insulin to allow the body to use its glucose for energy and metabolism. The blood glucose levels of diabetics are often elevated above normal values. Long term affects of this can include vision loss, kidney failure, and neurological damage.

An automatic insulin delivery system, or artificial pancreas, could help reduce the long term effects by providing a more natural, gradual injection of insulin based on the body's needs. One thing to be kept in mind is that the control is strictly one way, the injection of insulin. Naturally, this means that our controller will be ineffective against hypoglycemia, which is when blood glucose is at an abnormally low level. The only robustness that the artificial pancreas can guard against is to make sure that the controller does not result in or send the person into hypoglycemia. However, it is hoped that any device which is constantly monitoring blood glucose would at least provide some warning before hypoglycemia is achieved. For further information on automatic control strategies, see [Hod94].

We will only be simulating external disturbances of increased blood glucose to evaluate the performance of our controllers, since the only response to a disturbance which lowers glucose, like exercise, would be for the controller to turn off. This effect is readily seen in the simulations when glucose levels drop or are brought close to the desired level.

7.4 Control Implementation

7.4.1 Equilibrium. To use the nonlinear dynamics, we must first do a coordinate transformation so that each transformed state settles at zero. Letting q be the equilibrium value we will use the substitution

$$x = q + \delta x \tag{7.16}$$

Differentiating Eqn. (7.16) gives

$$\dot{x} = \delta\dot{x} \tag{7.17}$$

Equilibrium will depend on the health factors α and β. Since we are interested in control of a diabetic's blood glucose, we shall use in Eqn. (7.16) the diabetic equilibrium values given in Table 7.4. These equilibrium values were determined numerically from the final steady state of an undisturbed simulation.

Table 7.4 Diabetic Equilibrium Values

q_{cl}	10.928
q_{el}	70.034
q_{ggl}	22.898
q_{gl}	104.79
q_{ghl}	2.3730
q_{il}	6.4219
q_{gngl}	2.6169e-4
q_{is}	3.0597e+4
q_{gs}	1.5917e+4

7.4.2 Pseudo-Linearization. To implement a nonlinear controller would require either the calculations to be done in real time, or for a large lookup table to be implemented. Since the model has 9 states, neither is highly desirable. For this reason we will examine a reduced model. Since linearizing the glucose state was the initial problem, we might ask what happens if we keep only nonlinear terms that are a function of x_{gl}? If this model could capture the dynamics reasonably well, then a one dimensional lookup table would be easily implementable. Linear interpolation would be much quicker and require less memory than a 9 space tensor.

Since we wish to have only nonlinear terms involving only δx_{gl}, we will use Taylor series expansions on any nonlinear functions of the other states and remove higher order terms. The nonlinearities are mostly contained in the function trigh. Expanding and keeping linear terms we have

$$\text{trigh}(x,l,h) \simeq \text{trigh}(q,l,h) + \left.\frac{d\text{trigh}(x,l,h)}{dx}\right|_q \delta x \tag{7.18}$$

We will also have nonlinear terms that involve simple multiplication of two states. An example would be a term such as $x_{gl}x_{il}$ becoming $q_{gl}q_{il} + q_{gl}\delta x_{il} + \delta x_{gl}q_{il} + \delta x_{gl}\delta x_{il}$. The last term is kept because its nonlinearity involves δx_{gl}, whereas something like $x_{ghl}x_{il}$ reduces to only $q_{ghl}q_{il} + q_{ghl}\delta x_{il} + \delta x_{ghl}q_{il}$, having thrown out the term $\delta x_{ghl}\delta x_{il}$. The second example never occurs in our nonlinear equations.

The partially linearized equations, detailed in the next section, respond to an external glucose disturbance as shown in Figure 7.3. These dynamics match the nonlinear dynamics only for the initial response. Therefore we must be careful in our implementation. The differences in responses is due to a build up of error as the states are propagated foward. Since we will be measuring the states and calculating the derivatives based on those values, we do not have to worry about the error accumulating. Provided that the derivatives calculated at each time step have relatively small error, we can reasonably base a controller on these estimates of the derivatives.

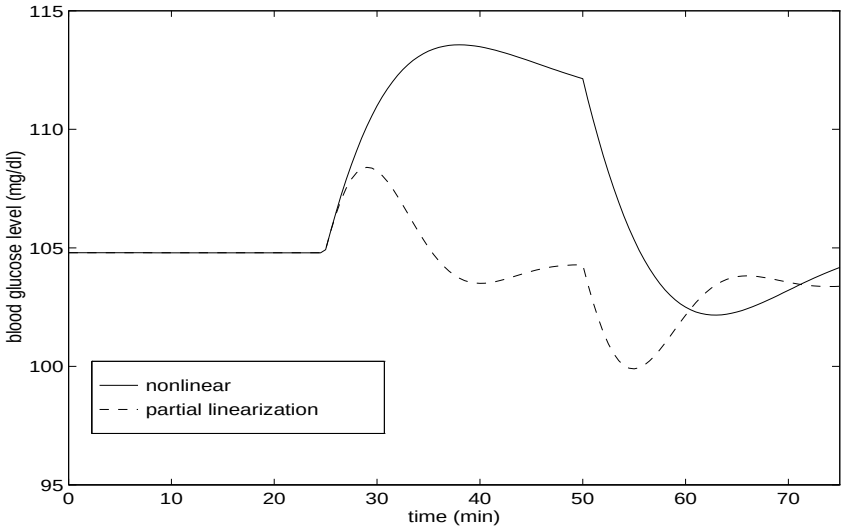

Figure 7.3 Response of Nonlinear Dynamics and Partially Linearized Dynamics

7.4.3 SDC Parameterization. After performing the coordinate transformation and pseudo-linearization, we are left with only one parameterization by allowing $A(x)$ to be a function of only δx_{gl}. Any constant terms will have to be multiplied by $\frac{\delta x_{gl}}{\delta x_{gl}}$, where the numerator will be factored

out leaving the coefficient for the δx_{gl} state. In the following equations we will denote in bold the δx that are factored out leaving the elements of our nonlinear matrix $A(x_{gl})$. The final form can be seen in Appendix A. The next chapter will detail the results of designing controllers for this model.

Our perturbed nonlinear equations become:

$$\delta \dot{x}_{cl} = -c_{cl}\delta \mathbf{x_{cl}} + \left[-c_{cl}q_{cl} + c_{cl}b_{cl}\frac{1+\text{trigh}(q_{gl}+\delta x_{gl},90,60)}{1+\text{trigh}(b_{gl},90,60)}\right]\frac{\delta \mathbf{x_{gl}}}{\delta x_{gl}} \tag{7.19}$$

$$\delta \dot{x}_{el} = -c_{el}\delta \mathbf{x_{el}} + \left[-c_{el}q_{el} + c_{el}b_{el}\frac{\text{trigh}(q_{gl}+\delta x_{gl},1.2b_{gl},0.8b_{gl})}{\text{trigh}(b_{gl},1.2b_{gl},0.8b_{gl})}\right]\frac{\delta \mathbf{x_{gl}}}{\delta x_{gl}} \tag{7.20}$$

$$\begin{aligned}\delta \dot{x}_{ggl} = & -c_{ggl}\delta \mathbf{x_{ggl}} + \\ & \left[-c_{ggl}q_{ggl} + 0.1(1.0-\beta)c_{ggl}b_{ggl} + \frac{0.4q_{gs}\text{trigh}(q_{gl}+\delta x_{gl},140\tilde{\alpha},60\tilde{\alpha})}{1000\cdot vol}\right]\frac{\delta \mathbf{x_{gl}}}{\delta x_{gl}} \\ & + \frac{0.4\text{trigh}(q_{gl}+\delta x_{gl},140\tilde{\alpha},60\tilde{\alpha})}{1000\cdot vol}\delta \mathbf{x_{gs}}\end{aligned} \tag{7.21}$$

$$
\begin{aligned}
\delta\dot{x}_{gl} &= \frac{1}{10\cdot vol}\left[24\cdot 60\cdot 2mass\cdot \delta\mathbf{x_{gngl}} + \frac{0.91\cdot 2mass}{2\text{trigh}(1.0,0.9,5.0)-\text{trigh}(1.0,0.25,5.0)}\cdot\right.\\
&\left(\frac{d\text{trigh}(\frac{q_{el}}{b_{el}},0.9,5.0)}{dx}\delta\mathbf{x_{el}} + \frac{d\text{trigh}(\frac{q_{ggl}}{b_{ggl}},0.9,5.0)}{dx}\delta\mathbf{x_{ggl}}\right)\\
&-\left(\frac{0.91\cdot 2mass}{2\text{trigh}(1.0,0.9,5.0)-\text{trigh}(1.0,0.25,5.0)}\cdot\frac{d\text{trigh}(\frac{q_{il}}{b_{il}},0.25,5.0)}{dx} - \frac{0.5mass}{b_{il}}\right)\cdot\\
&\left.\left(\frac{q_{gl}+\delta x_{gl}}{b_{gl}}\right)\delta\mathbf{x_{il}}\right]\\
&+\left[24\cdot 60\cdot 2mass\cdot q_{gngl} - 0.75\cdot 2mass\frac{\text{trigh}(q_{gl}+\delta x_{gl},60,150)}{\text{trigh}(b_{gl},60,150)}\right.\\
&+0.91\cdot 2mass\frac{\text{trigh}(\frac{q_{el}}{b_{el}},0.9,5.0)+\text{trigh}(\frac{q_{ggl}}{b_{ggl}},0.9,5.0)-\frac{q_{gl}+\delta x_{gl}}{b_{gl}}\text{trigh}(\frac{q_{il}}{b_{il}},0.25,5.0)}{2\text{trigh}(1.0,0.9,5.0)-\text{trigh}(1.0,0.25,5.0)}\\
&\left.-0.5mass\frac{(q_{gl}+\delta x_{gl})q_{il}}{b_{gl}b_{il}} - 47.999232+\right]\frac{\delta\mathbf{x_{gl}}}{10\cdot vol\cdot\delta x_{gl}} + \frac{w_{\text{glucose}}}{10\cdot vol} \qquad (7.22)
\end{aligned}
$$

$$
\delta\dot{x}_{ghl} = -c_{ghl}\delta\mathbf{x_{ghl}} + \left[-c_{ghl}q_{ghl} + c_{ghl}b_{ghl}\frac{1+\text{trigh}(q_{gl}+\delta x_{gl},90,60)}{1+\text{trigh}(b_{gl},90,60)}\right]\frac{\delta\mathbf{x_{gl}}}{\delta x_{gl}} \qquad (7.23)
$$

$$
\begin{aligned}
\delta\dot{x}_{il} &= -c_{il}\delta\mathbf{x_{il}} + \left[-c_{il}q_{il} + \frac{0.4q_{is}\text{trigh}(q_{gl}+\delta x_{gl},60,140)}{1000\cdot vol}\right]\frac{\delta\mathbf{x_{gl}}}{\delta x_{gl}}\\
&+\frac{0.4\text{trigh}(q_{gl}+\delta x_{gl},60,140)}{1000\cdot vol}\delta\mathbf{x_{gs}} + \frac{u_{\text{insulin}}}{vol} \qquad (7.24)
\end{aligned}
$$

$$
\begin{aligned}
\delta\dot{x}_{gngl} \;=\;& -c_{gngl}\delta\mathbf{x_{gngl}} + \\
& \left[-c_{gngl}q_{gngl} + \frac{c_{gngl}b_{gngl}}{2\text{trigh}(1.0,1.0,5.0)+2\text{trigh}(1.0,0.9,5.0)-\text{trigh}(1.0,0.5,5.0)}\cdot\right. \\
& \left[\text{trigh}(\frac{q_{cl}}{b_{cl}},1.0,5.0)+\text{trigh}(\frac{q_{el}}{b_{el}},0.9,5.0)+\text{trigh}(\frac{q_{ggl}}{b_{ggl}},0.9,5.0)\right. \\
& \left.\left.+\ \text{trigh}(\frac{q_{ghl}}{b_{ghl}},1.0,5.0)-\text{trigh}(\frac{q_{il}}{b_{il}},0.5,5.0)\right]\right]\frac{\delta\mathbf{x_{gl}}}{\delta x_{gl}} \\
& +\frac{c_{gngl}b_{gngl}}{2\text{trigh}(1.0,1.0,5.0)+2\text{trigh}(1.0,0.9,5.0)-\text{trigh}(1.0,0.5,5.0)}\cdot \\
& \left[\frac{d\text{trigh}(\frac{q_{cl}}{b_{cl}},1.0,5.0)}{dx}\delta\mathbf{x_{cl}}+\frac{d\text{trigh}(\frac{q_{el}}{b_{el}},0.9,5.0)}{dx}\delta\mathbf{x_{el}}+\frac{d\text{trigh}(\frac{q_{ggl}}{b_{ggl}},0.9,5.0)}{dx}\delta\mathbf{x_{ggl}}\right. \\
& \left.+\ \frac{d\text{trigh}(\frac{q_{ghl}}{b_{ghl}},1.0,5.0)}{dx}\delta\mathbf{x_{ghl}}-\frac{d\text{trigh}(\frac{q_{il}}{b_{il}},0.5,5.0)}{dx}\delta\mathbf{x_{il}}\right] \qquad (7.25)
\end{aligned}
$$

$$
\begin{aligned}
\delta\dot{x}_{gs} \;=\;& \left[-0.2\alpha-0.4\text{trigh}(q_{gl}+\delta x_{gl},140\tilde{\alpha},60\tilde{\alpha})\right]\delta\mathbf{x_{gs}} \\
& +\left[0.2\alpha(m_{gs}-q_{gs})-0.4q_{gs}\text{trigh}(q_{gl}+\delta x_{gl},140\tilde{\alpha},60\tilde{\alpha})\right. \\
& \left.-0.1(1.0-\beta)c_{ggl}b_{ggl}\cdot 1000vol\right]\frac{\delta\mathbf{x_{gl}}}{\delta x_{gl}} \qquad (7.26)
\end{aligned}
$$

$$
\begin{aligned}
\delta\dot{x}_{is} \;=\;& \left[-0.2\beta-0.4\text{trigh}(q_{gl}+\delta x_{gl},60,140)\right]\delta\mathbf{x_{is}} \\
& +\left[0.2\beta(m_{is}-q_{is})-0.4q_{is}\text{trigh}(q_{gl}+\delta x_{gl},60,140)\right]\frac{\delta\mathbf{x_{gl}}}{\delta x_{gl}} \qquad (7.27)
\end{aligned}
$$

VIII. State Regulation of Blood Glucose

This chapter presents the results of the nonlinear quadratic control as applied to human glucose and insulin dynamics. This is a tracking problem, since we wish the glucose levels to be stabilized at a level different than a diabetic's basal (average equilibrium) level. Since the controller dynamics are based on deviations away from equilibrium, those values must be subtracted from the measured states. This setup can be seen in Figure 8.1. To regulate glucose, instead of the

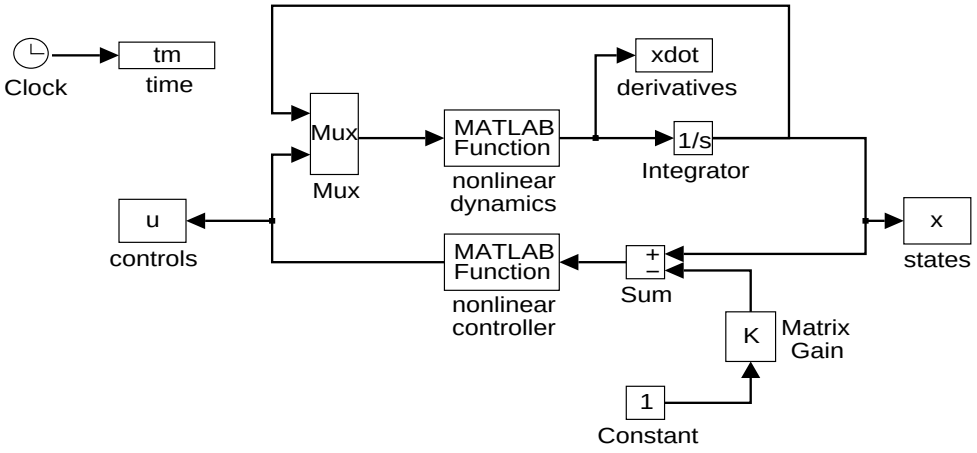

Figure 8.1 SIMULINK Tracking Diagram

equilibrium value being subtracted, we will subtract the desired level causing the regulator to be a tracker. The block 'Matrix Gain' is the vector containing those equilibrium and tracking values. The functions used for the 'nonlinear dynamics' and 'nonlinear controller' blocks can be seen in Appendix A.

8.1 Continuous Controller Solution

We will examine regulation of blood glucose level by penalizing only the glucose state since it is the objective of our regulation problem. However, it is important not to penalize it too much

relative to insulin injection (i.e. control usage), since large doses of insulin and fast rate changes in hormone levels are both dangerous.

We should also leave the other states unpenalized for another reason. With the injection of insulin and change in blood glucose level, the other states will depart from their diabetic basal levels as well. This effect is necessary, and we do not wish to further reduce our controller effectiveness by penalizing those deviations. It might be possible to find a desirable level for those other states and weight deviations from the desired levels, but this is risky and could cause a situation where glucose will be driven below its desired value for some disturbance, resulting in hypoglycemia.

All the simulations in this section use the partially linearized equations developed in the previous chapter. To evaluate the performance of the nonlinear quadratic regulator, we will examine the response to a disturbance in blood glucose. The external disturbance will start at 25 minutes and last until 50 minutes which will represent the intake of glucose from food.

To develop a feasible controller, we are first interested in verifying that our SDC parameterization works, and that a solution to regulate glucose exists. To examine this we will first set up a cheap control problem. Using as much control as is needed, we will see if the controller is effective.

The penalty matrices for the cheap control problem are:

$$Q = \text{diag}\left(\begin{bmatrix} 0 & 0 & 0 & 10^8 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right) \tag{8.1}$$

$$R = 1 \tag{8.2}$$

Figures 8.2 and 8.3 shows the cheap controller results of NQR in response to the external glucose injection (i.e. food). The controller is indeed quite effective in regulating glucose level. There is a slight steady state error because of the tracking nature of the problem, which requires control usage as long as glucose is away from its real equilibrium. Also, there is an initial insulin

injection with magnitude $5 \cdot 10^4$ which is not plotted in Figure 8.3 to allow the magnitudes of the rest of the control history to show.

This only establishes that there is a possible solution. We now must work at trying to achieve a more realistic controller. First, negative insulin is not implementable, so our control usage must be limited to only positive values. Second, the control usage magnitude is too large and would be fatal. For our next try, we will add a limiter preventing negative insulin injection and use weights of:

$$Q = \text{diag}\left(\begin{bmatrix} 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right) \tag{8.3}$$

$$R = 1 \tag{8.4}$$

The results are seen in Figures 8.4 and 8.5. Here we see that glucose is not very effectively controlled, and it reaches levels almost as high as the uncontrolled diabetic would achieve. The problem is that the insulin injection still starts with a peak that is too high and dangerous. We cannot achieve the performance we want. To get better glucose tracking we need to increase the state weighting, but that would only increase the insulin injection peaks to more dangerous levels.

We will now examine two different methodologies to back away from the strict use of NQR. First, we are interested in what happens if we also limit the maximum insulin injection rate. If an artificial pancreas was implemented, it would certainly have a failsafe limit far below the maximum injection a person could handle. Using an upper limit of 20 mU/s, and choosing a weight on glucose of 10^6 keeping $R = 1$, we get the response seen in Figures 8.6 and 8.7. Note: this is not implying this is a safe or healthy limit at all. This is merely the maximum value used in [Hod94], and is used solely for a proof of concept and comparison to earlier work. In fact, Hodel states it is ten times the natural basal injection level of humans.

This is definitely the most interesting control history of the cases we will examine. Each narrow pulse has a duration of 0.5 seconds. The controller is using almost bang-bang control. In

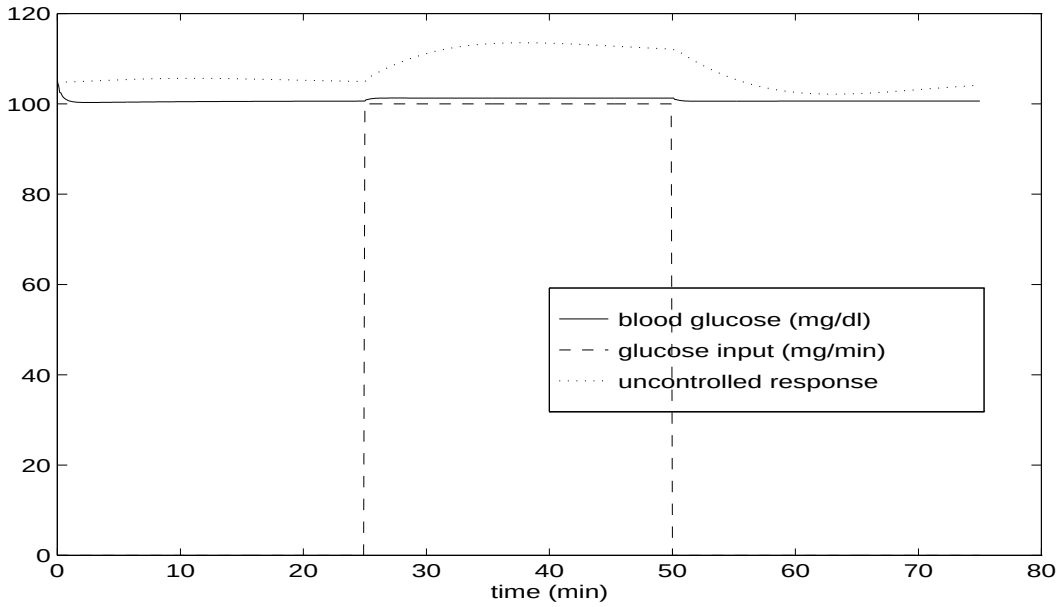

Figure 8.2 Glucose Dynamics for the Cheap Control Problem

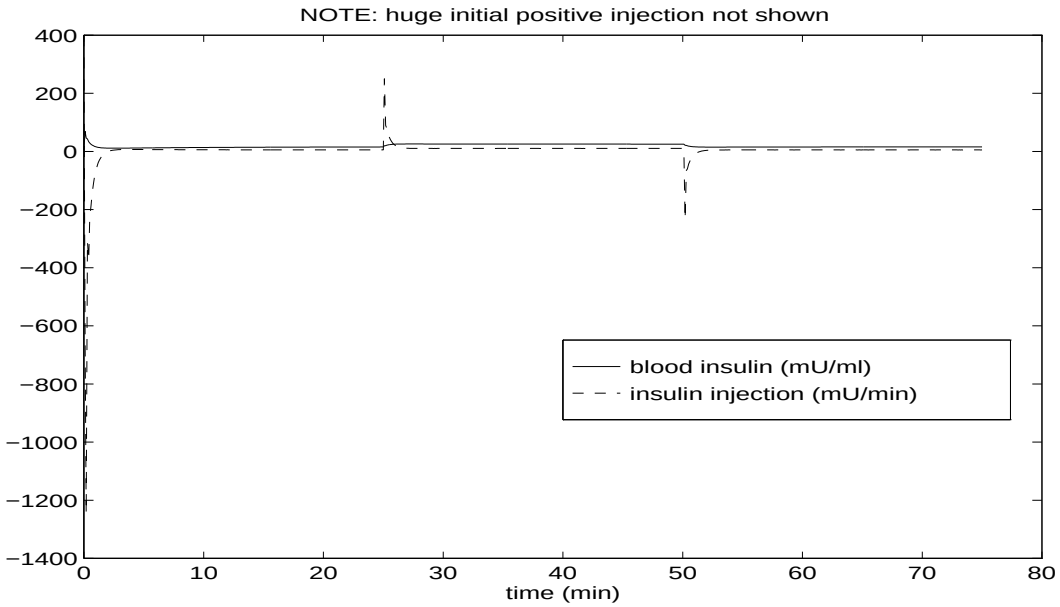

Figure 8.3 Insulin Dynamics for the Cheap Control Problem

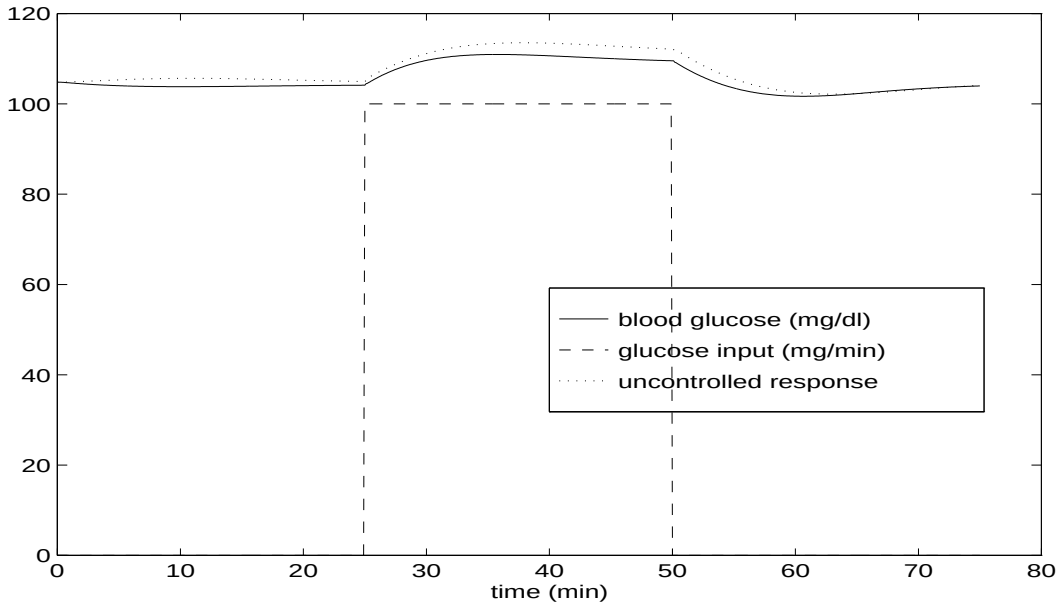

Figure 8.4 Glucose Dynamics for $Q = 100$

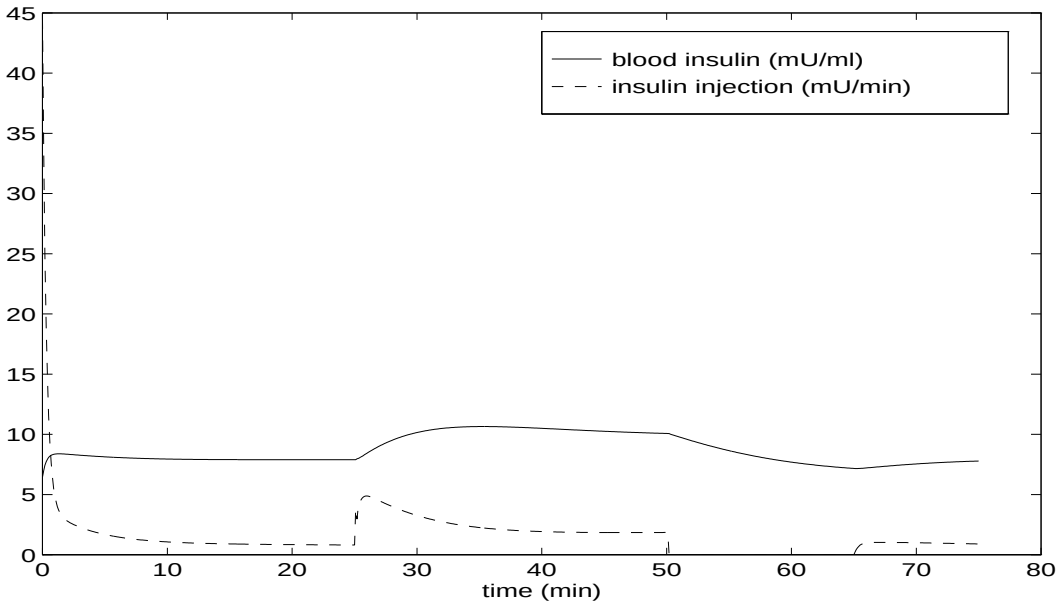

Figure 8.5 Insulin Dynamics for $Q = 100$

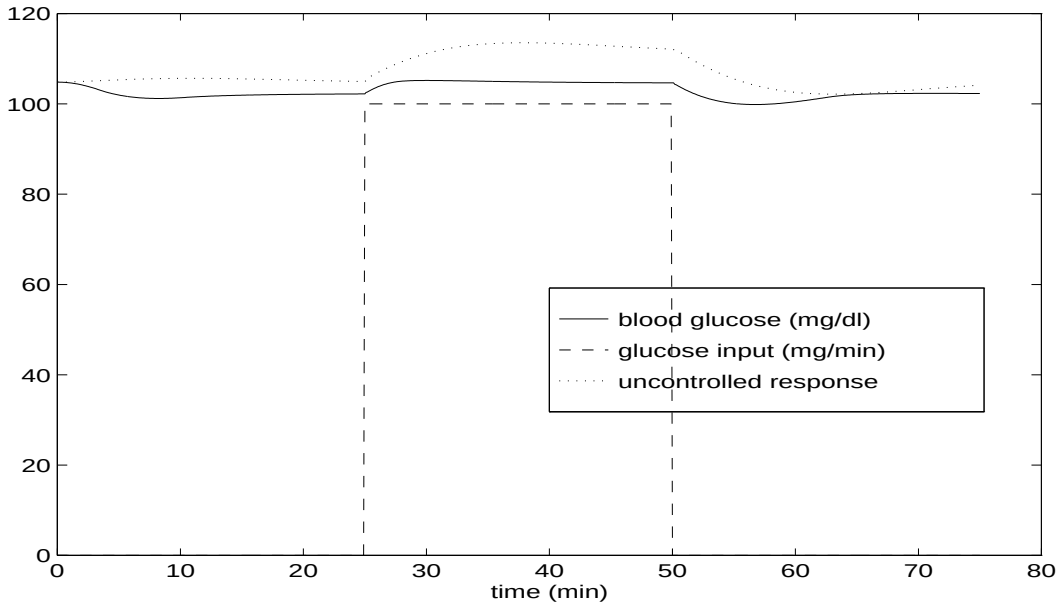

Figure 8.6 Glucose Dynamics for Limited Magnitude Controller

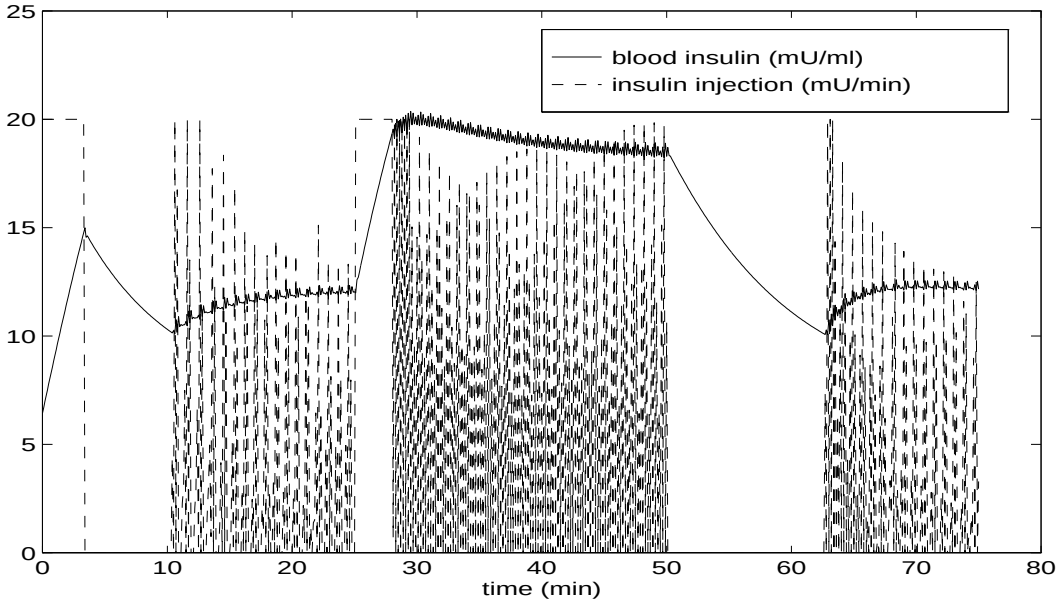

Figure 8.7 Insulin Dynamics for Limited Magnitude Controller

fact, increasing the weighting on glucose another order of magnitude does result in strictly bang-bang control. Here the effects are slightly less severe. The results are promising and blood glucose is regulated to acceptable levels; however, the changes in control usage can be very drastic. With further research and validation of the assumed dynamics, this control methodology may provide one possible solution.

We will also examine one more control strategy to see if we can establish a smoother control history. Here we will return to a cheap control problem for weighting, except instead of implementing the commanded insulin injection we will scale back the control by a constant value that would result in safe levels. This is not always an acceptable control strategy, since stability is not guaranteed when changing the gain of a calculated controller. However, for this problem the dynamics are well behaved, and we know a reduction in control usage will only slow the rates of insulin injection. We are not in any more risk of inducing hypoglycemia than with the fully effective controller. The weight on glucose for this problem is 10^8 with $R = 1$. The implemented control will be

$$u = \frac{u_{\text{commanded}}}{2500} \tag{8.5}$$

Notice in Figures 8.8 and 8.9 that the control history is much more well behaved. Glucose levels are regulated to reasonable levels and insulin injections are under the maximum we used in the previous design.

To show the effectiveness of this controller, Figure 8.10 shows the controlled diabetic response against a healthy response to the same disturbance. The match is quite close. We can see that the healthy response is not trying to keep a flat response, but that the natural dynamics allow gradual increases and decreases in response to disturbances, just as the controlled response does.

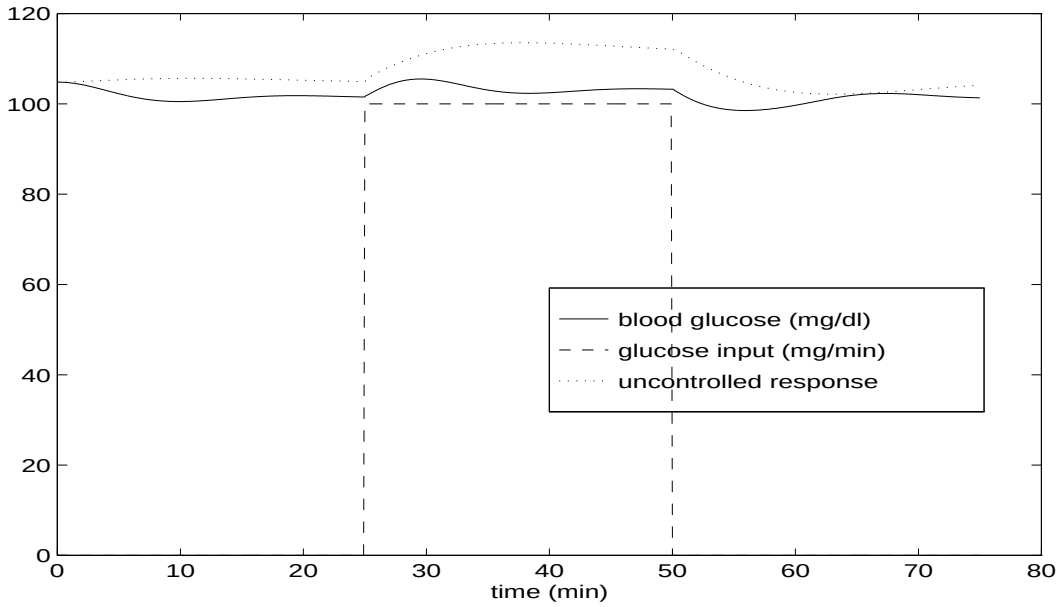

Figure 8.8 Glucose Dynamics for Reduced Usage Controller

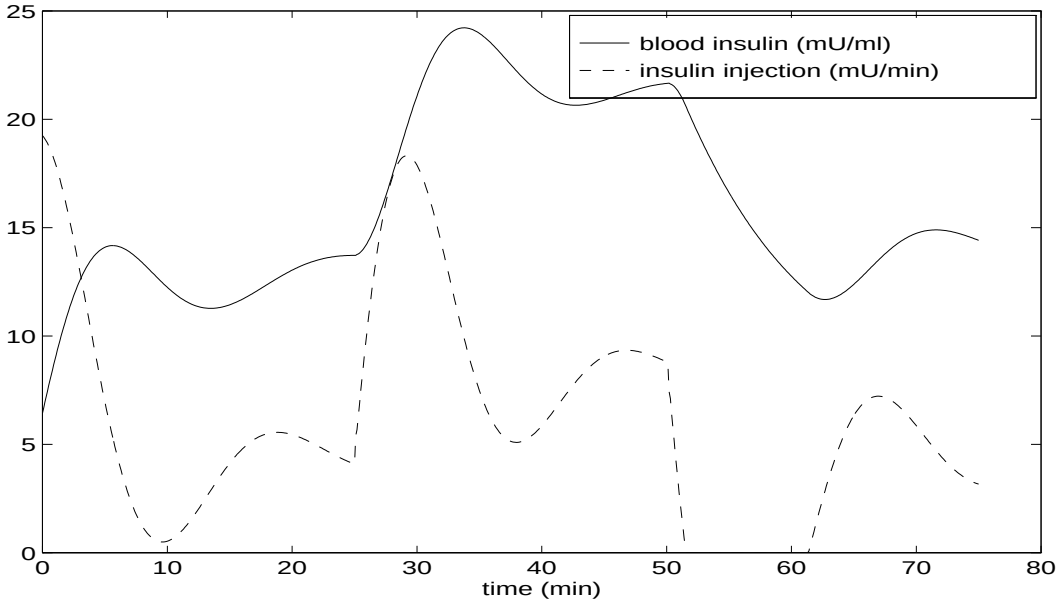

Figure 8.9 Insulin Dynamics for Reduced Usage Controller

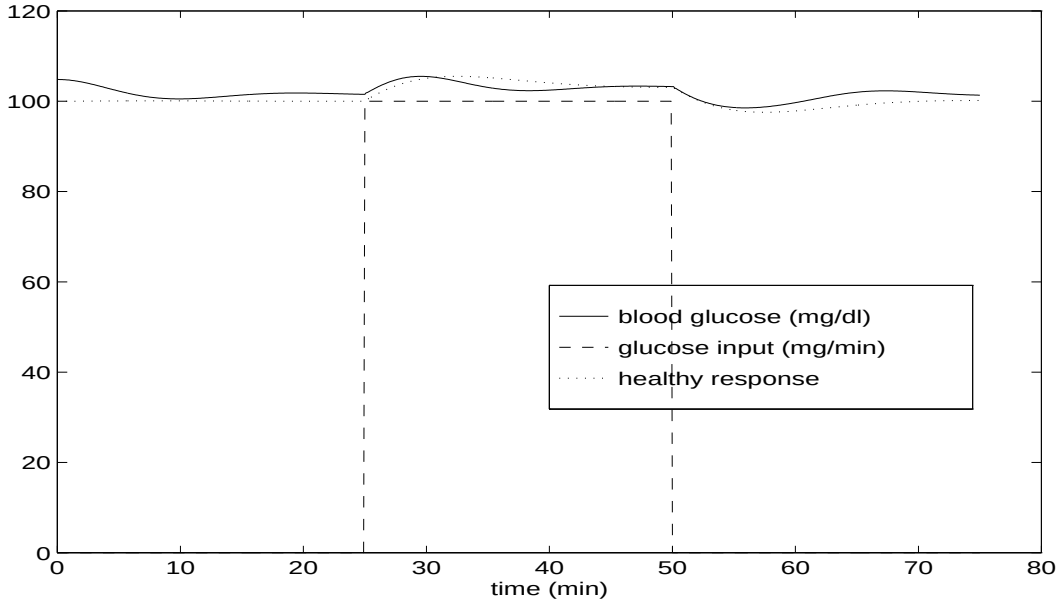

Figure 8.10 Controlled Diabetic Response vs. Healthy Response

8.2 Table Lookup Solution

The pseudo-linearization was done to form simpler calculations which could be done off-line and stored in a table lookup. In this section, the controller uses a gain matrix based on glucose measurements to the nearest 0.5 mg/dl. Whereas the previous section used a continuous solution, here the solution P is solved for a δx_{gl} from 0.0 to 20.0 at steps of 0.5. The elements of P are then stored as a row in a larger matrix. Instead of calculating the solution to the SDARE at each step, δx_{gl} is used to calculate the index to the matrix. The matrix P is reformed and used to calculate the control usage.

The results for this methodology are given in Figures 8.11 and 8.12. The results are indistinguishable from the pointwise solution of above. However, the simulation ran at least twice as fast and provides a more implementable controller. Instead of having an onboard processor to solve an algebraic Riccati equation, an artificial pancreas can use the gain matrices which would be stored in memory.

8.3 Control without Full State Feedback

Since NQR assumes full state feedback, none of the controllers examined for blood glucose regulation are actually implementable. Presently, blood glucose is the only measurement that would be available. Additionally, two of the states, namely glucose stores and insulin stores, are not measurable at all. They are results of the model, and there would be no way to measure total amounts in the body.

Perhaps the other states' influences on control usage are relatively small compared to blood glucose. To see if this might be reasonable, we will set the gains for all other states to 0 and calculate the control usage. Figures 8.13 and 8.14 shows the results of this methodology. Insulin injection achieves higher values but the deviations on blood glucose remain reasonable. Scaling the control usage again may be effective in bringing the insulin injection levels back to reasonable

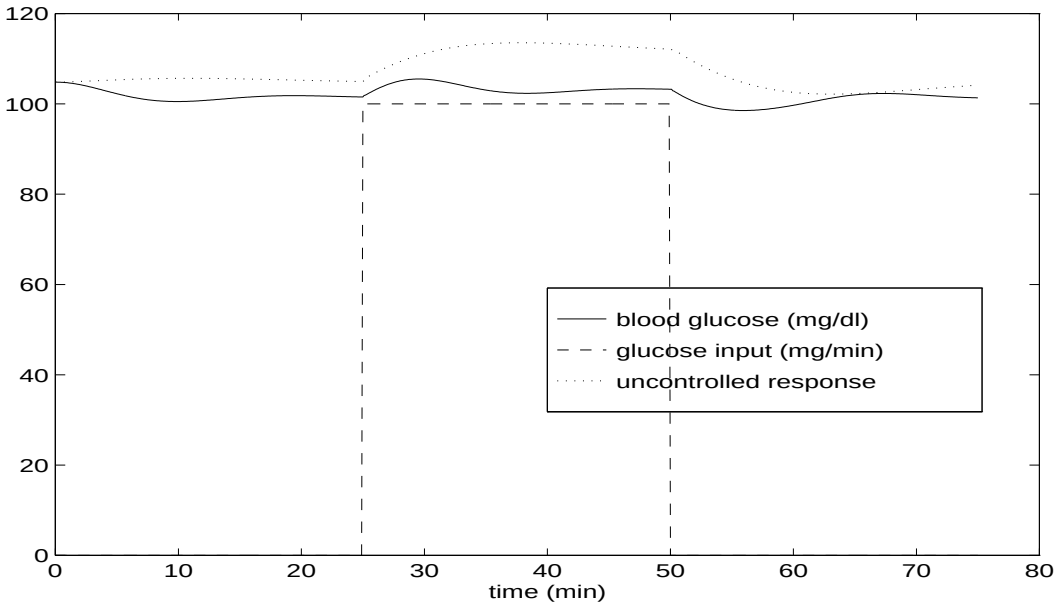

Figure 8.11 Glucose Dynamics for Discretized Controller

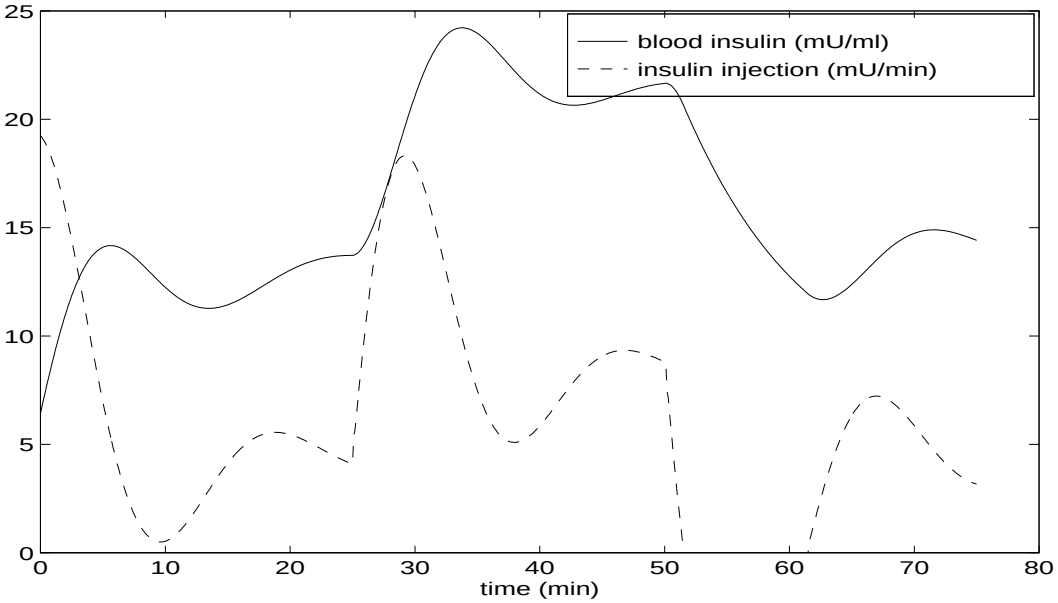

Figure 8.12 Insulin Dynamics for Discretized Controller

levels. Scaling the control usage by half gives the results in Figures 8.15 and 8.16. The control usage is indeed brought back to reasonable levels and the glucose level is still well controlled for this disturbance.

These results also make the possibility of incorporating a nonlinear state estimation encouraging. Since removing the gains on the other states did not have a completely damaging effect, a nonlinear estimator that can converge on reasonable state estimations may be effective in a combined regulator-estimator controller. We could then leave all the gains intact and use the estimated states with small errors in the states having little impact. Here we must caution against using the partially linearized dynamics for the purpose of state estimation. Since this implementation does propagate our assumed dynamics forward, the steady state is significantly different than steady state of the full nonlinear dynamics. If we wish to couple a state estimator with our controller we must use a full nonlinear estimator for this purpose.

8.4 Performance to Large Disturbances

Diabetics can be prone to very high blood glucose levels which exceed the levels already examined. The previous simulations were run with the same disturbance used in Naylor, et al. [NHS95], to allow comparisons. This section will use the final controller, i.e. discretized with scaled control usage $u = \frac{u_{\text{commanded}}}{5000}$, examined in the previous section with a maximum injection rate of 20mU/min. We will increase the external glucose disturbance to be 0.5 grams/min over the same period from 25 to 50 minutes. This level was chosen because the model achieved a high blood glucose level 125 mg/dl for a healthy response, which is near the upper safety limit of blood glucose. Note the diabetic response reaches an uncontrolled high of 173 mg/dl, which is dangerous when continued for prolonged periods of time. Since the AEMG model is still in an early stage of development, this test case might be stretching the model. However, we are interested in how NQR can handle increasingly larger disturbances. For linear controllers, large disturbances can cause the

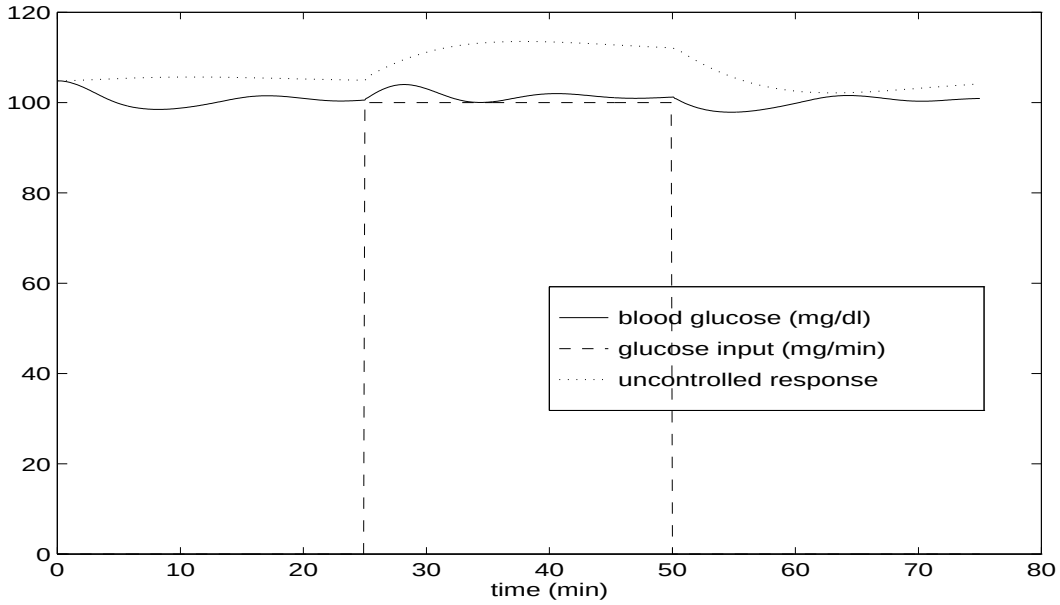

Figure 8.13 Glucose Dynamics: Non x_{gl} gains set equal to zero

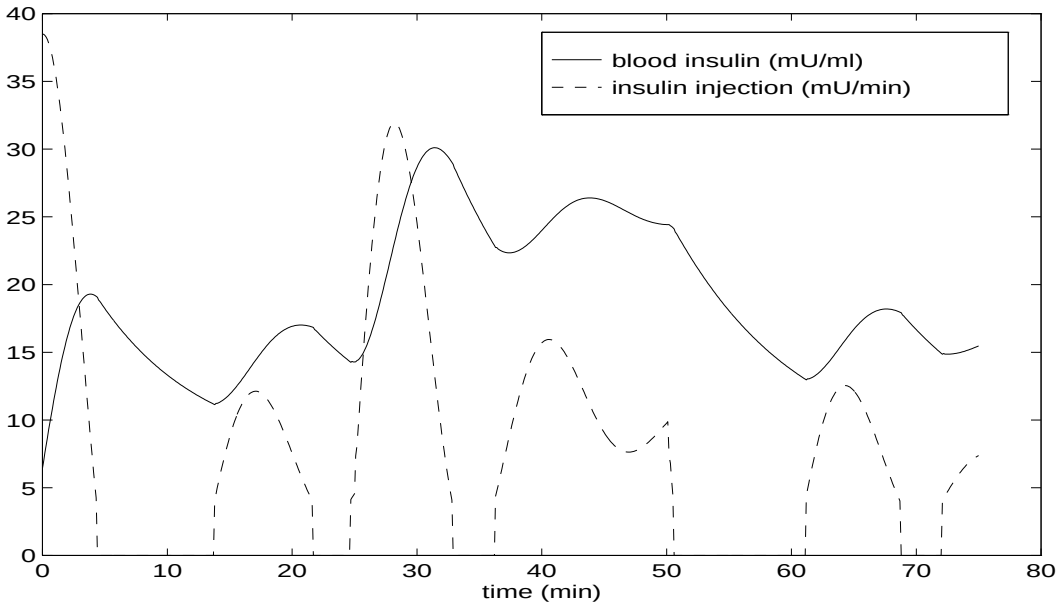

Figure 8.14 Insulin Dynamics: Non x_{gl} gains set equal to zero

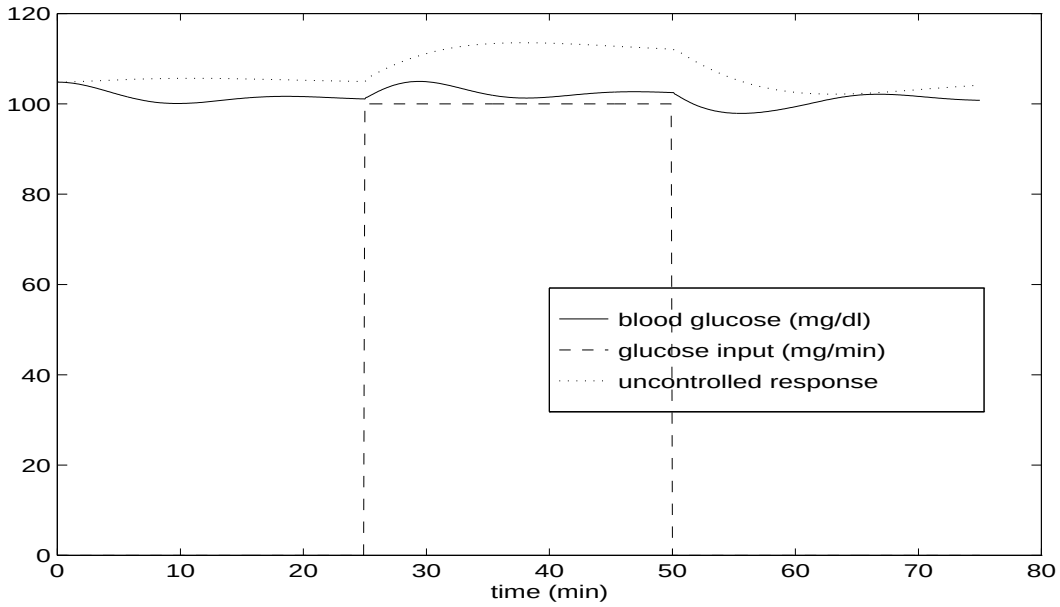

Figure 8.15 Glucose Dynamics: Zeroed gains with reduced control

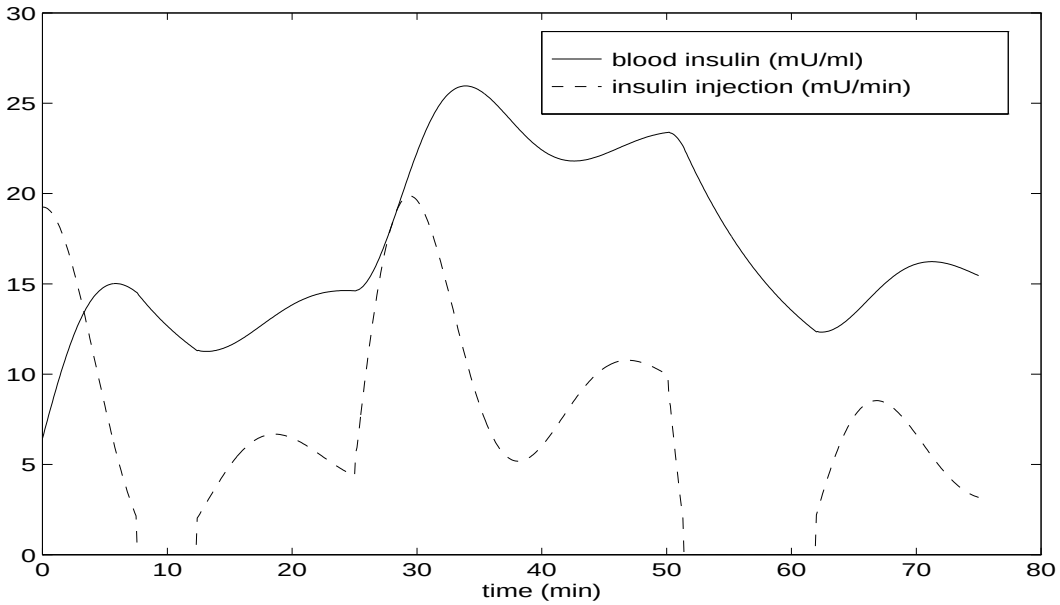

Figure 8.16 Insulin Dynamics: Zeroed gains with reduced control

controller to be ineffective if the dynamics are highly nonlinear. If the NQR controller can continue to provide encouraging results to increasingly large disturbances, then it seams reasonable that a nonlinear quadratic regulator could provide one solution for an artificial pancreas.

The results are shown in Figures 8.17 and 8.18. The controlled response performs very closely to the healthy response. This near match is of course unique for this case. For larger disturbances the controller will result in lower performance than the average healthy performance, due to the upper safety limit on insulin injection. Since the controller shown here is using the maximum injection level, it will not be able to achieve a higher blood insulin level. For smaller disturbances, the behavior approaches that seen in Figure 8.15.

Figure 8.19 shows how the controller would behave if its control usage was not limited to a maximum value, with the corresponding control history in Figure 8.20. This shows that even though the the control usage may not implementable, mathematically the NQR controller can keep up with the dynamics of the nonlinear system. The overshoots increase for large disturbances, but as a regulator NQR does return to the equilibrium values.

8.5 Summary

This chapter shows a preliminary investigation into a control methodology for an artificial pancreas. While the model of human biological dynamics may be at an unvalidated stage, we were still able to see nonlinear quadratic control effectively handle the assumed dynamics. Where previous linear controllers have provided discouraging results, the SDARE calculated a suitable controller. As the model improves, we can hope that the controller would improve as well.

On a more cautionary note, we must remind ourselves that full state feedback is not available for this problem. Unfortunately, whereas we were able to turn off the gains to the other states and still achieve good results, as the model improves this may not be the case. However, these initial

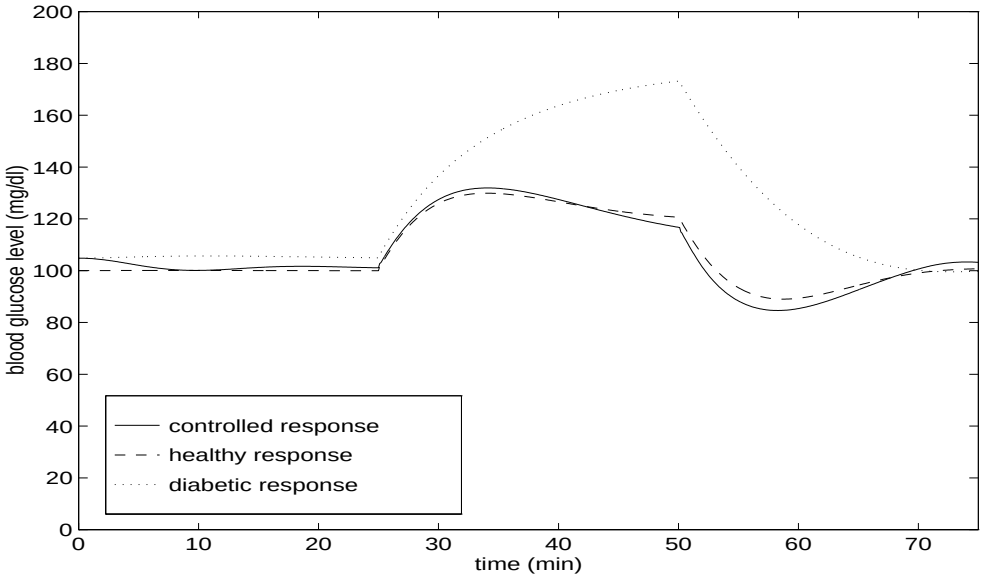

Figure 8.17 Glucose Dynamics: 0.5 g/min disturbance

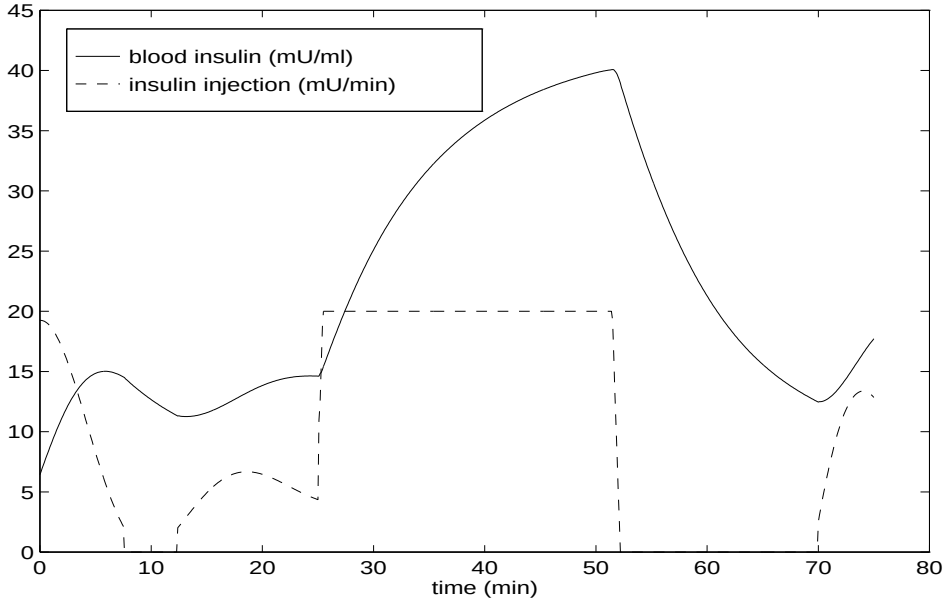

Figure 8.18 Insulin Dynamics: 0.5 g/min disturbance

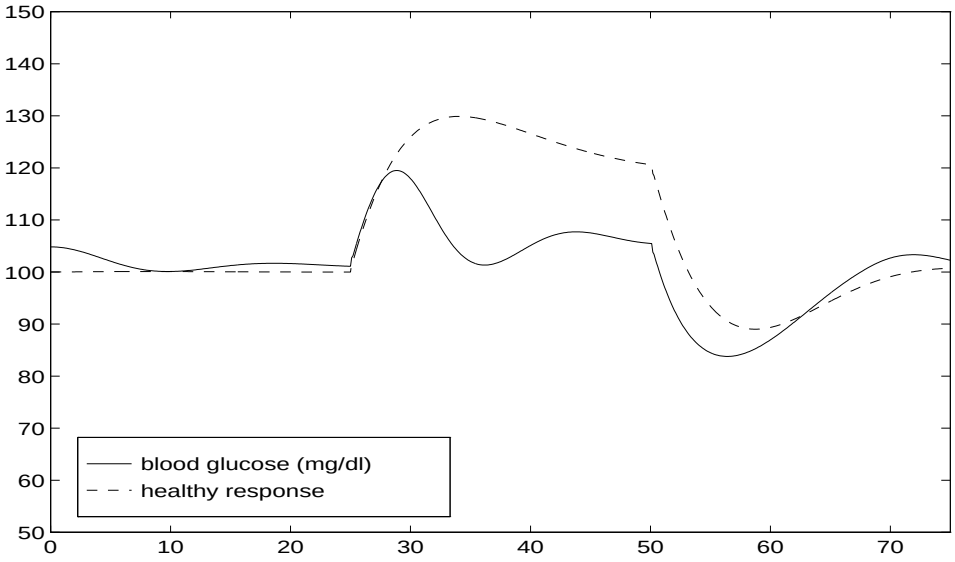

Figure 8.19 Glucose Dynamics: 0.5 g/min disturbance, no insulin cutoff

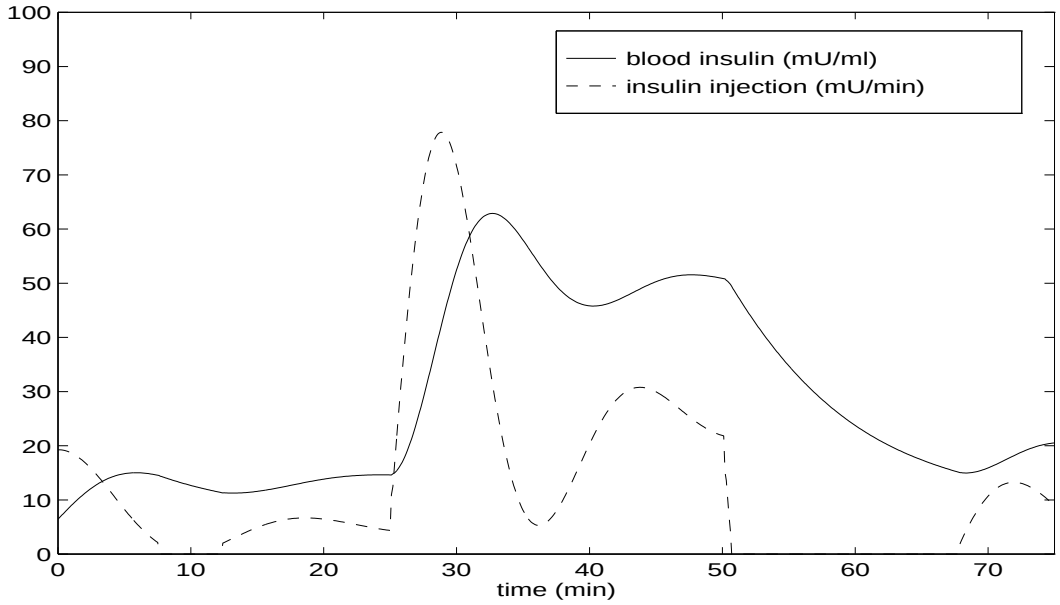

Figure 8.20 Insulin Dynamics: 0.5 g/min disturbance, no insulin cutoff

results should encourage us to pursue nonlinear estimation for this problem if we indeed need all the states.

IX. Conclusions and Recommendations

9.1 Further Research Areas

This thesis provided only a first attempt at applying the methods of Cloutier, et al., to practical applications. There is still further research required in each of the areas we have examined.

9.1.1 Internally Stabilized Satellites. Nonlinear quadratic regulation provided some very encouraging results when applied to a satellite stabilized by internal momentum wheels. NQR provides both a method for stabilizing the satellite and reorienting it to any desired position. Further research in this area should probably investigate the merits of NQR as compared to controllers already being implemented.

Because the equations of motions were scaled, the method should be examined using real values for the inertias. This will determine both if the torques are of reasonable magnitude, and if the time to settle is acceptable. As the inertias represented by the scaled inertia matrix go up, the control magnitudes and time scale likewise increase.

9.1.2 Artificial Pancreas Studies. This thesis only provided a proof of concept in examining the applicability of NQR as a basis for artificial pancreas control. While the results are quite encouraging, there is still much more research required on the modelling side. Not only do the biological dynamics need to be well understood, but they also require rigorous validation before implementing a control based on them.

As the dynamic models of glucose, insulin, and related hormones improve, it is this author's hope that a controller based on the state dependent coefficient method of NQR will continue to be a viable solution. It is most likely that as the equations improve, they will become only more complex and more nonlinear, implying that nonlinear controllers may hold the most promising solutions.

Further research is not limited to the biological side. There are still many issues of NQR to be examined, including robustness and using nonlinear estimators. There is the very real possibility

that for an improved dynamics model the gains on the other states will be significant. Since NQR is a full state feedback method, this will require it being coupled with some type of state estimator. This could have disastrous effects on stability and robustness. Ultimately, since the controller would be implemented on a human, there is no room for casual guessing.

9.1.3 Gain Scheduling Alternative. One possibility for NQR that the pancreas study provided is the use of the state dependent coefficient form as an alternative to gain scheduling. Linear controllers of nonlinear systems have been adequate for many different control problems where the dynamics are reasonably represented and the perturbations relatively small. Perhaps implementing SDC controllers on partially linearized systems will likewise be adequate for systems with nonlinearities contained mostly in one state. This would provide a bridge between linear and full-up nonlinear controllers. A follow up study could examine the problem where the dynamics are represented by

$$\dot{x} = A(x_k)x + B(x_k)u \tag{9.1}$$

where x_k is the single state in which all nonlinearities are expressed. Perhaps there are more easily derived optimality conditions and robustness issues for this simpler problem.

9.1.4 Discrete Implementation. What is important to remember is that the simulations presented provided instantaneous control usage based on the values of the states. Although this helps provide a proof of concept, it does not confirm that we can implement the controller automatically in a delayed or digitally sampled manner. Stability is provided asymptotically by providing a pointwise stabilizing controller. If we implement the calculated control usage at any delayed time, there is no guarantee that the commanded control will still have a stabilizing effect. This is of course true for linear systems as well.

None of this should discourage one from pursuing a discrete implementation. Further research should also look at the performance of sampling and delay in a more realistic implementation of NQR.

9.2 Summary

While the theory of nonlinear feedback control using the state-dependent Riccati equation is still relatively new, it appears that there may be great promise in the results. For the examples studied in this thesis, NQR provided stabilizing controllers that were quite effective, and although suboptimal still provided adequate state and control usage responses.

We have also seen nonlinear quadratic regulation used on two very different dynamical systems. The method seems well suited to a variety of control problems. With further development of the theory behind NQR, we may find this method to be a promising new direction in control theory.

Appendix A. MATLAB Implementation of NQR

This appendix provides the functions which created the nonlinear dynamics and controllers used in the SIMULINK simulation.

A.1 Rigid Body Dynamics

The following MATLAB function generates the equations of motion for the externally controlled satellite.

```
function rtrn=satdyn1(x,J)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create nonlinear satellite dynamics
% angular velocity and quaternion states
% due to external torques
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

w=x(1:3); q=x(4:6); q4=x(7); T=x(8:10);


wdot= -inv(J)*crs(w)*J*w +inv(J)*T;
qdot=  .5*(crs(q)+q4*eye(3))*w;
q4dot= -.5*q'*w;

rtrn= [ wdot; qdot; q4dot];
```

A.2 Internal Rotor and Satellite Dynamics

The following MATLAB function generates the equations of motion for the externally controlled satellite.

```
function rtrn=satdyn2(z,J,A)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create nonlinear satellite dynamics with internal rotors
% states: mu, x, q; control: u
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mu=z(1:3); x=z(4:6); q=z(7:9); q4=z(10); u=z(11:13);

mudot=  u;
w=      inv(J)*(x-A*mu);
xdot=   crs(x)*w;
qdot=   0.5*(crs(q)+q4*eye(3))*w;
```

```
q4dot= -0.5*q'*w;

rtrn= [mudot;xdot;qdot;q4dot];
```

A.3 Artificial Pancreas

The following MATLAB function generates the dynamics used in the AEMG model with some minor modifications.

```
function dx = nlplant(u, alpha, beta)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% artificial pancreas dynamics
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xcl=  u(1);
xel=  u(2);
xggl= u(3);
xgl=  u(4);
xghl= u(5);
xil=  u(6);
xgngl=u(7);
xgs=  u(8);
xis=  u(9);

EGI=  u(10); %external glucose
EII=  u(11); %external insulin

d1= -log(.5)/7.5;
d2= -log(.5)/90;
d3= -log(.5)/30;

vol=6; mass=80;

as=1.1-.1*beta;

bcl=    11.1;
bel=    92;
bggl=  120;
bgl=   100;
bghl=    2.4;
bil=    35;
bgngl=    .00027083;
bgs=  5*d1*bggl*vol*1000;
bis=  5*d1*bil*vol*1000;

upgs= .4*xgs*trigh([xgl as*140 as*60 ]) + .1*(1-beta)*d1*bggl*vol*1000;
upis= .4*xis*trigh([xgl 60 140]);
uhgp= 24*60*2*mass*xgngl + .7*(1.3)*2*mass*(trigh([xel/bel .9 5])...
```

```
 + trigh([xggl/bggl .9 5]) - trigh([xil/bil .25 5])*xgl/bgl)...
 /(2*trigh([1 .9 5]) - trigh([1 .25 5]));
utgcu=.75*2*mass*trigh([xgl 60 150])/trigh([bgl 60 150])...
 + xgl*xil*mass/2/bgl/bil;

maxgluc= 10*d1*bggl*vol*1000;
maxins= 10*d1*bil*vol*1000;

dxcl= -d1*xcl + d1*bcl*(1+trigh([xgl 90 60]))/(1+trigh([bgl 90 60]));
dxel= -d1*xel + d1*bel*trigh([xgl 1.2*bgl .8*bgl])*2;
dxggl= -d1*xggl + upgs/1000/vol;
dxgl= (uhgp - utgcu + EGI - 47.999232)/10/vol;
dxghl= -d2*xghl + d2*bghl*(1+trigh([xgl 90 60]))/(1+trigh([bgl 90 60]));
dxil= -d1*xil +  upis/1000/vol + EII/vol;
dxgngl= -d3*xgngl + d3*bgngl*(trigh([xcl/bcl 1 5]) + trigh([xel/bel .9 5])...
  + trigh([xggl/bggl .9 5]) + trigh([xghl/bghl 1 5]) - trigh([xil/bil .5 5]))...
  /(2*trigh([1 1 5]) + 2*trigh([1 .9 5]) - trigh([1 .5 5]));
dxgs= .2*alpha*(maxgluc - xgs) - upgs;
dxis= .2*beta*(maxins - xis) - upis;

dx=[dxcl dxel dxggl dxgl dxghl dxil dxgngl dxgs dxis]';
```

A.4 Satellite Controller using External Torques

The following MATLAB function generates the feedback control for the externally controlled satellite.

```
function torq=cont1(x,J,R,Q)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% nonlinear controller for satellite w/ external torques
% using reduced dynamics
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

w=x(1:3); q=x(4:6); q4=x(7);

B=  [inv(J);zeros(3)];

A=  [ -inv(J)*crs(w)*J,      zeros(3);...
       (crs(q)+q4*eye(3))/2, zeros(3)];

P= sdare(A, B*inv(R)*B', Q);

torq= -inv(R)*B'*P*x(1:6);
```

A.5 Satellite Controller of Internal Momentum Wheels

The following MATLAB function generates the feedback control for the internally controlled satellite.

```
function rtrn=cont2(z,J,A,s,r)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% nonlinear controller for intenally stabilized satellite
% dimension 13, states: mu, x, w, q; control: u
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mu=z(1:3); x=z(4:6); q=z(7:9); q4=z(10);

Q= diag([0 0 0 0 0 0 s s s s s s 0]);
R= eye(3)*r;

w=inv(J)*(x-A*mu);
JxJ=inv(J)*crs(x)*inv(J);
qmat= [.5*(crs(q)+q4*eye(3)); -.5*q' ];

B=[eye(3); zeros(3); -inv(J)*A; zeros(4,3)];

A=[ zeros(3,13)                          ;...
    zeros(3,6),   crs(x),  zeros(3,4) ;...
    -JxJ*A ,      JxJ,     zeros(3,7) ;...
    zeros(4,6),   qmat,    zeros(4)   ];

P=sdare(A, B*inv(R)*B', Q);

%eig(A)                %uncomment to see eigenvalues of A
%eig(A-B*inv(R)*B'*P)  %uncomment to see eigenvalues of (A-BK)

rtrn= -inv(R)*B'*P*[mu;x;w;q;q4];
```

A.6 Partially Linearized Pancreas Controller

The following MATLAB function generates the feedback control for the artificial pancreas model.

```
function  u=nlcontr(x,Q,R)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% artificial pancreas controller
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

y=x(4); %glucose state
```

```
if abs(y)<.0001
u=0;

else
B= [0 0 0 0 0 1/6 0 0 0]';
A= nlA(y); %create nonlinear A matrix

P= are(A, B*inv(R)*B', Q);

u= -inv(R)*B'*P*x;
% u= u/2500;  % uncomment for scaled controller
end
```

The above code calls the function $nlA(x_{gl})$ which is given by:

```
function aa = nlA(xgl)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Make nonlinear A matrix; function of xgl only
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

d1= -log(.5)/7.5;
d2= -log(.5)/90;
d3= -log(.5)/30;
vol= 6; mass= 80; alpha= 0.1; beta= 0.1;
v10= vol*10;
as= 1.1-0.1*beta;

bcl=    11.1;
bel=    92;
bggl=  120;
bgl=   100;
bghl=    2.4;
bil=    35;
bgngl=   .00027083;
bgs=  5*d1*bggl*vol*1000;
bis=  5*d1*bil*vol*1000;

qcl=    10.83;
qel=    70.24;
qggl=   23.868;
qgl=   104.82;
qghl=    2.3814;
qil=     6.43;
qgngl=   2.6353e-4;
qgs=     3.1885e4;
qis=     1.591e4;


upgs= 0.4*qgs*trigh([qgl+xgl as*140 as*60 ]) + 0.1*(1-beta)*d1*bggl*vol*1000;
upis= 0.4*qis*trigh([qgl+xgl 60 140]);
gama1= 0.7*(1.3)*2*mass/(2*trigh([1 0.9 5]) - trigh([1 0.25 5]));
```

```
gama2= d3*bgngl/(2*trigh([1 1 5]) + 2*trigh([1 0.9 5]) - trigh([1 0.5 5]));

maxgluc= 10*d1*bggl*vol*1000;
maxins= 10*d1*bil*vol*1000;

a14= -d1*qcl + d1*bcl*(1 + trigh([qgl+xgl 90 60]))/(1+trigh([bgl 90 60]));

a24= -d1*qel + d1*bel*trigh([qgl+xgl 1.2*bgl 0.8*bgl])*2;

a34= -d1*qggl + upgs/1000/vol;
a38= .4*trigh([qgl+xgl as*140 as*60])/1000/vol;

a41= gama1*dtr(qel/bel,0.9,5)/v10;
a43= gama1*dtr(qggl/bggl,0.9,5)/v10;
a44= 24*60*2*mass*qgngl + gama1*(trigh([qel/bel 0.9 5])...
  + trigh([qggl/bggl 0.9 5]) - trigh([qil/bil 0.25 5])*(qgl+xgl)/bgl);
a44= (a44 - (0.75*2*mass*trigh([qgl+xgl 60 150])/trigh([bgl 60 150])...
  + (qgl+xgl)*qil*mass/2/bgl/bil + 47.999232))/v10;
a46= (-gama1*dtr(qil/bil,.25,5) - (qgl + xgl)*mass/(2*bil*bgl))/v10;
a47= 24*60*2*mass/v10;

a54= -d2*qghl + d2*bghl*(1 + trigh([qgl+xgl 90 60]))/(1 + trigh([bgl 90 60]));

a64= -d1*qil + upis/1000/vol;
a69= .4*trigh([qgl+xgl 60 140])/1000/vol;

a71= gama2*dtr(qcl/bcl,1,5);
a72= gama2*dtr(qel/bel,0.9,5);
a73= gama2*dtr(qggl/bggl,0.9,5);
a74= -d3*qgngl + gama2*(trigh([qcl/bcl 1 5]) + trigh([qel/bel 0.9 5])...
  + trigh([qggl/bggl 0.9 5]) + trigh([qghl/bghl 1 5]) - trigh([qil/bil 0.5 5]));
a75=  gama2*dtr(qghl/bghl,1,5);
a76= -gama2*dtr(qil/bil,0.5,5);

a84= 0.2*alpha*(maxgluc-qgs) - upgs;
a88= 0.2*alpha - 0.4*trigh([qgl+xgl as*140 as*60]);

a94= 0.2*beta*(maxins-qis) - upis;
a99= 0.2*beta  - 0.4*trigh([qgl+xgl 60 140]);

aa=[-d1   0    0  a14/xgl 0   0   0   0   0;...
      0  -d1   0  a24/xgl 0   0   0   0   0;...
      0   0   -d1 a34/xgl 0   0   0   a38 0;...
      a41 0   a43 a44/xgl 0   a46 a47 0   0;...
      0   0    0  a54/xgl -d2 0   0   0   0;...
      0   0    0  a64/xgl 0   -d1 0   0   0;...
      a71 a72 a73 a74/xgl a75 a76 -d3 0   0;...
      0   0    0  a84/xgl 0   0   0   a88 0;...
      0   0    0  a94/xgl 0   0   0   0   a99];
```

Appendix B. Reduction of Neutrally Stable System

This appendix shows how the 7 state model given in Eqn. (6.1) can be reduced to a fully controllable form such that a standard algebraic Ricatti equation solver can be used. We will assume that controls will be weighted by $R = \rho^2 \mathrm{I}$, the angular velocities weighted by $Q_\omega = \gamma^2 \mathrm{I}$, and the first three quaternions by $Q_{q_{1-3}} = \zeta^2 \mathrm{I}$

Beginning with the SDARE we have

$$\begin{bmatrix} J\omega^{\times}J^{-1} & \mathcal{Q}^{\mathrm{T}} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} P_{11} & P_{12}^{\mathrm{T}} \\ P_{12} & P_{22} \end{bmatrix} + \begin{bmatrix} P_{11} & P_{12}^{\mathrm{T}} \\ P_{12} & P_{22} \end{bmatrix} \begin{bmatrix} -J^{-1}\omega^{\times}J & 0 \\ \mathcal{Q} & 0 \end{bmatrix} - \frac{1}{\rho^2} \begin{bmatrix} P_{11} & P_{12}^{\mathrm{T}} \\ P_{12} & P_{22} \end{bmatrix} \begin{bmatrix} J^{-1}J^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} P_{11} & P_{12}^{\mathrm{T}} \\ P_{12} & P_{22} \end{bmatrix} + \begin{bmatrix} \gamma^2 \mathrm{I}_{3\times 3} & 0 \\ 0 & \begin{bmatrix} \zeta^2 \mathrm{I}_{3\times 3} & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix} = 0 \tag{B.1}$$

Expand each term to get

$$\begin{bmatrix} J\omega^{\times}J^{-1}P_{11} + \mathcal{Q}^{\mathrm{T}}P_{12} & J\omega^{\times}J^{-1}P_{12}^{\mathrm{T}} + \mathcal{Q}^{\mathrm{T}}P_{22} \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -P_{11}J^{-1}\omega^{\times}J + P_{12}^{\mathrm{T}}\mathcal{Q} & 0 \\ -P_{12}J^{-1}\omega^{\times}J + P_{22}\mathcal{Q} & 0 \end{bmatrix} - \frac{1}{\rho^2} \begin{bmatrix} P_{11}J^{-1}J^{-1}P_{11} & P_{11}J^{-1}J^{-1}P_{12}^{\mathrm{T}} \\ P_{12}J^{-1}J^{-1}P_{11} & P_{12}J^{-1}J^{-1}P_{12}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \gamma^2 \mathrm{I}_{3\times 3} & 0 \\ 0 & \begin{bmatrix} \zeta^2 \mathrm{I}_{3\times 3} & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix} = 0 \tag{B.2}$$

We now have the following three equations to solve for P_{11}, P_{12}, and P_{22}:

$$J\omega^{\times}J^{-1}P_{11} + \mathcal{Q}^{\mathrm{T}}P_{12} - P_{11}J^{-1}\omega^{\times}J + P_{12}^{\mathrm{T}}\mathcal{Q} - \frac{1}{\rho^2}P_{11}J^{-1}J^{-1}P_{11} + \gamma^2\mathrm{I} = 0_{3\times 3} \quad \text{(B.3)}$$

$$J\omega^{\times}J^{-1}P_{12}^{\mathrm{T}} + \mathcal{Q}^{\mathrm{T}}P_{22} - \frac{1}{\rho^2}P_{11}J^{-1}J^{-1}P_{12}^{\mathrm{T}} = 0_{3\times 4} \quad \text{(B.4)}$$

$$-\frac{1}{\rho^2}P_{12}J^{-1}J^{-1}P_{12}^{\mathrm{T}} + \begin{bmatrix} \zeta^2\mathrm{I}_{3\times 3} & 0 \\ 0 & 0 \end{bmatrix} = 0_{4\times 4} \quad \text{(B.5)}$$

Solving for P_{12} first we take one term across and taking effectively the square root of each side with careful attention to matrix dimension

$$P_{12}J^{-1} = \rho\zeta \begin{bmatrix} \mathrm{I}_{3\times 3} \\ 0 \end{bmatrix} \quad \text{(B.6)}$$

where each side is 4×3. Postmultiply by J to get

$$P_{12} = \rho\zeta \begin{bmatrix} J \\ 0 \end{bmatrix} \quad \text{(B.7)}$$

This establishes that the last row of P_{12} is a zero row. Now examining Eqn. (B.4) and expanding

$$J\omega^{\times}J^{-1}\begin{bmatrix} J & 0 \end{bmatrix} + \mathcal{Q}^{\mathrm{T}}P_{22} - \frac{1}{\rho^2}P_{11}J^{-1}J^{-1}\begin{bmatrix} J & 0 \end{bmatrix} = 0_{3\times 4} \quad \text{(B.8)}$$

we see that the first and third terms have a fourth column whos elements are all zero. This implies that P_{22} must have a fourth zero column and row because of symmetry. Therefore, the seventh row and column of $P(x)$ which solves the SDARE are empty. We can then eliminate the seventh row and column from our dynamics and we are left with a strictly controllable parameterization. The equations for P_{11} reduce to another SDARE.

Bibliography

CDM95. J. R. Cloutier, C. N. D'Souza, and C. P. Mracek. Nonlinear Regulation and Nonlinear H_∞ Control Via the State-Dependent Ricatti Equation Technique. *Submitted for publication*, 1995.

Cho91. V. A. Chobotov. *Spacecraft Attitude Dynamics and Control.* Krieger Publishing Company, 1991.

Hal95a. C. D. Hall. Asst Professor, Dept of Aero & Astro, Air Force Institute of Technology, WPAFB, OH. Personal communication, 1995.

Hal95b. C. D. Hall. Spinup Dynamics of Gyrostats. *Journal of Guidance, Control, and Dynamics*, 18(5):1177–1183, 1995.

Hod94. A. S. Hodel. Automatic Control Issues in the Development of an Artificial Pancreas. *Internal Auburn University Report*, August 1994.

MAT. MATLAB: High performance numeric computation and visualization software. The Math Works, Inc., Natick MA, 1994.

MCD95. C. P. Mracek, J. R. Cloutier, and C. N. D'Souza. A New Technique for Nonlinear Estimation. *Submitted for publication*, 1995.

Nay94. J. S. Naylor. Automatic Control Issues in the Development of an Artificial Pancreas. *Internal Auburn University Report*, August 1994.

NHS95. J. S. Naylor, A. S. Hodel, and D. Schumacher. Automatic Control Issues in the Development of an Artificial Pancreas. In *Proceedings of the American Control Conference*, pages 771–775, Seattle, WA, June 1995.

ZDG95. K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control.* Prentice Hall, 1995.

Vita

Captain David K. Parrish was born March 24, 1968, in Newcastle, Wyoming. He obtained his Bachelor of Science in Aerospace Engineering from Arizona State University in May, 1991. While awaiting active duty he substitute taught in the Scottsdale and Mesa School Districts in Arizona. Captain Parrish started active duty in April, 1992 in the Flight Controls and Stability home office at Wright-Patterson AFB, Ohio. After six months of training, he transferred to the Training Systems Program Office in Aeronautical Systems Command. Captain Parrish was responsible for certification of various simulator handling qualities, including C-141, T-1A, and the B-1B. In May 1994, Captain Parrish began his Masters Degree Program at the Air Force Institute of Technology.

Permanent address: 255 E. Warwick Ave.
Newcastle, Wyoming 82701
email: DKParrish@aol.com